FACULTAT D'INFORMÀTICA DE BARCELONA

Gestió d'activitats esportives amb client Android i servidor Drupal

Projecte Final de Carrera - Memòria final

Alumne Jordi Cabeza López

Director (Secretari) Borja Vallés Fuente

President Juan Luis Esteban Ángeles

Codirector Àlex Ballarin

Vocal Teresa Monreal Arnal

18/01/2011

Índex

1 Introducció

1.1 Context del projecte

En els següents apartats es donaran a conèixer les motivacions, objectius, característiques i abast del projecte desenvolupat. Aquesta memòria està basada en el desenvolupament del mateix.

El projecte va néixer de la necessitat d'ampliar una plataforma web existent, utilitzada com a prova de concepte d'una tecnologia (Drupal) enfocada a la gestió de turisme rural. Aquesta plataforma es vol aprofitar per provar una nova tecnologia (Android) permetent la comunicació entre les dues tecnologies, Drupal en el rol servidor i Android en el rol de client. Es van definir dos desenvolupaments paral·lels independents, un per la millora de la gestió dels allotjaments rurals i un altre per desenvolupar la gestió d'activitats i, en particular, de partides de golf amb la creació del client per a dispositius mòbils. Aquest projecte a estat destinat a realitzar el segon desenvolupament.

D'aquesta manera, van sorgir diferents propostes per millorar el sistema actual, algunes de les quals van acabar formant el projecte que es presentarà en aquest document.

1.1.1 Objectiu del projecte

L'objectiu del projecte és dissenyar i implementar una plataforma client/servidor basada en dos tecnologies de codi obert conegudes, com ara Drupal pel servidor web, i Android per les aplicacions de dispositius mòbils. La comunicació entre ambdós es farà amb una interfície amb tecnologia estàndard Web Services/JSON .

Com s'ha comentat, el projecte neix d'una plataforma ja existent. En aquesta plataforma la gestió de continguts web relacionat amb activitats esportives té una presencia pràcticament testimonial que es redueix a la creació d'activitats per part de l'administració del sistema web mostrant una petita selecció amb les últimes activitats creades a la pantalla principal del projecte.

En aquest projecte s'ha intentat millorar la gestió d'activitats desenvolupada en l'anterior versió,enfocada especialment en la gestió de partides de golf, ja que és un bon exemple per explotar la interacció amb el client mòbil. També un objectiu del projecte és millorar la usabilitat del sistema així com facilitar la navegació en el lloc web sempre tenint en compte que en un futur, aquest projecte podrà tornar a ser evolucionat, de manera que els futurs desenvolupadors puguin utilitzar com a base la feina realitzada sense que aquesta feina sigui un impediment alhora d'afegir o millora certes funcionalitats.

En un camp més personal, l'objectiu del projecte és conèixer i agafar experiència tant en la plataforma del sistema web com la del client per a dispositius mòbils ja que són dos plataformes que es poden considerar madures i que tenen un gran potencial en el mon empresarial. També resultarà molt profitós agafar experiència en la planificació de projectes amb metodologies àgils, que estan en plena expansió, aprenent Scrum.

1.1.2 Característiques del projecte i situació actual

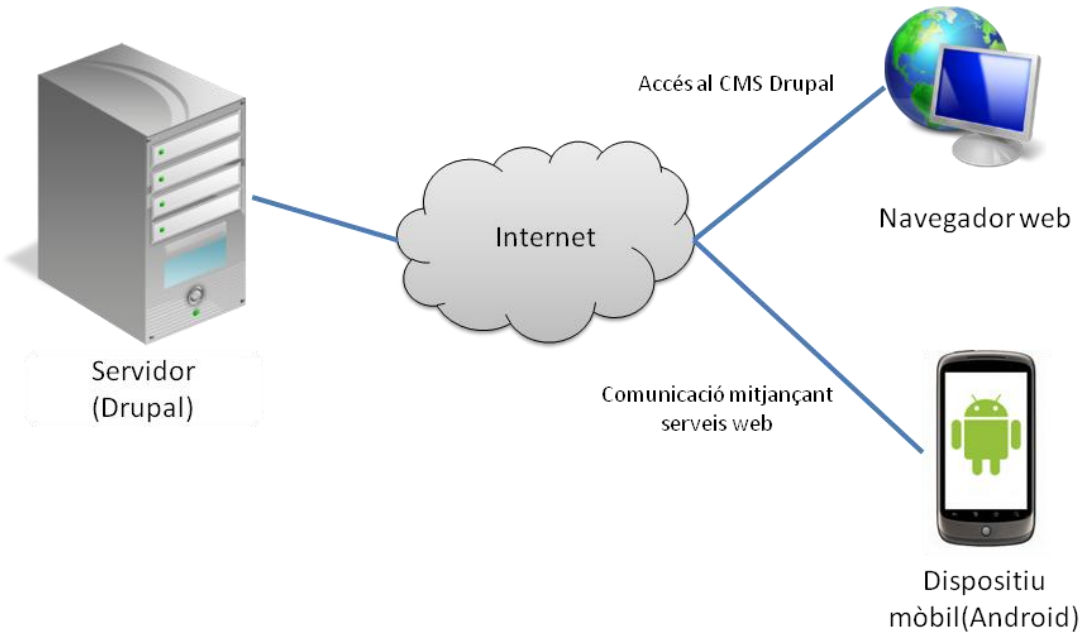
Aquest projecte és un projecte sense ànim de lucre i sense cap client real orientat a conèixer millor les tecnologies de codi obert que van madurant al mercat, com Drupal i Android]. Per això es va pensar en utilitzar un client fictici que estigues interessat en disposar d'una eina amb les funcionalitats que ofereix la plataforma desenvolupada.

Partint del punt d'un sistema web que s'amplia el client podria ser una associació de turisme rural que tingués una forta vinculació amb empreses d'activitats esportives i lúdiques i que volgués donar informació sobre activitats esportives que es realitzen en ubicacions pròximes a les cases rurals. Hem pres el golf com a activitat bàsica del nostre client fictici, perquè representa molt bé la mecànica d'us que volem assolir: els usuaris podrien organitzar els seus partits amb el nostre sistema i transmetre les puntuacions directament des del camp. D'aquesta manera l'associació de cases rurals oferirà un servei més complet que atraurà potencialment a un segment d'usuaris amb un poder adquisitiu més elevat, els jugadors de golf.

Aquesta eina facilitaria als usuaris la feina de cercar activitats en ubicacions properes a les cases rurals facilitant que els clients tornin a utilitzar el sistema de lloguer al disposar de tota

la informació en un mateix entorn. Altrament, es permetrà als propis usuaris poder crear activitats esportives on la resta d'usuaris es puguin apuntar, centrant-nos com hem dit abans en l'exemple concret del golf. La idea seria atraure a una comunitat activa d'usuaris interessats en el món del golf que poguessin augmentar el volum de lloguers.

El client també voldria entrar en el creixent món de les aplicacions per a dispositius mòbils permetent accedir a part de la informació del sistema (en aquest desenvolupament la gestió de partides de golf) com a prova per, en cas de tenir èxit, ampliar les funcionalitats en futurs desenvolupaments.



Il·lustració 1 – Arquitectura bàsica del projecte a desenvolupar

En relació a la data d'entrega es va marcar com límit acabar el desenvolupament abans de finalitzar l'any, preferiblement al novembre, per poder posar en funcionament el projecte a principis de 2011. No es va parlar de temes econòmics ja que no existia un client real del projecte.

1.2 Abast del projecte

Com s'ha comentat abans, el projecte parteix d'una plataforma ja existent. Aquesta plataforma està basada en el sistema gestor de continguts (CMS) modular Drupal[1]. En el

[1] Més informació de Drupal a: http://drupal.org/

següent punt del document detallarem el funcionament bàsic d'aquest gestor però, de moment, ens quedarem amb el concepte que Drupal és una plataforma basada en mòduls, del quals uns formen part del nucli (permetent gestió de continguts, d'usuaris, ...) i la resta són opcionals que amplien la plataforma des del punt de vista funcional (p.e. creació d'una botiga online) o tècnic (p.e. interfície de serveis web). Així doncs, es tracta de seleccionar i configurar els mòduls existents que s'adaptin a les necessitat del projecte i de desenvolupar a mida els mòduls que permetin assolir funcionalitats específiques del projecte.

La plataforma ofereix moltes funcionalitats que es tornen a aprofitar en aquest desenvolupament però, per a l'extensió concreta de gestionar partides de golf i donar accés des d'una interfície de serveis web no hi ha suport funcional, fent que s'hagi de desenvolupar un mòdul funcional.

En el cas de les funcionalitats que s'han de realitzar en Drupal, haurem d'escollir si es poden realitzar amb els mòduls que formen part del nucli (core) del CMS, o bé, utilitzant mòduls creats per la comunitat Open Source o, en última instancia, creant mòduls a mida per satisfer les necessitats del projecte.

Per altra, banda el client sobre dispositiu mòbil Android que desenvoluparem en Android[2] s'haurà de realitzar integrament. No obstant, s'estudiarà l'existència de llibreries creades que ens puguin facilitar la feina per assolir les funcionalitats que detallarem en el següent punt.

1.2.1 Funcionalitats noves: taula comparativa

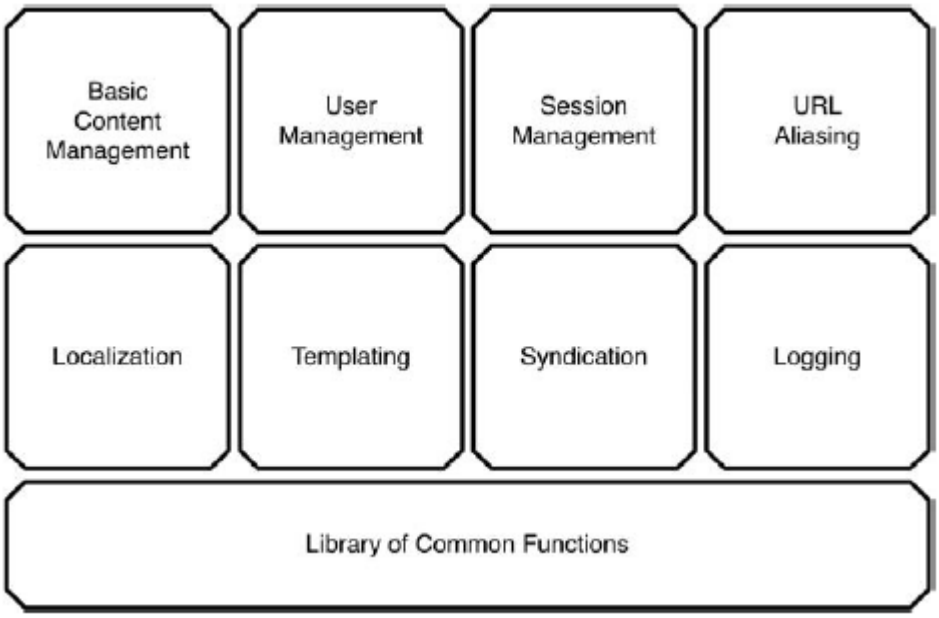
En la següent taula, indicarem les funcionalitats a desenvolupar sobre Drupal indicant en quin punt es troben abans de començar el nostre projecte en relació a versions prèvies del desenvolupament.

Diferenciarem el que porta Drupal de sèrie (mòduls core), el que se li pot afegir al Drupal si s'inclouen mòduls publicats pel intercanvi lliure a la web de la plataforma Drupal generats per altres programadors (http://drupal.org/project/modules), el desenvolupament original

[2] Més informació del sistema operatiu Android a: http://es.wikipedia.org/wiki/Android

de l'eina (Cases rurals v1) i el desenvolupament que es farà en aquest projecte (Cases rurals v2+ Activitats).

Abans de mostrar la taula, en la següent il·lustració podem veure les funcionalitats bàsiques que porta incorporades tot Drupal de sèrie per saber millor des de quin punt inicial es parteix del projecte.



Il·lustració 2 - Pro Drupal Development 2nd Edition – Funcionalitats més destacables que permeten els mòduls core de Drupal

Com es pot veure Drupal permet una gestió bàsica de continguts i de usuaris així com una gestió de les sessions dels usuaris o permetre la creació de plantilles per mostrar el contingut entre d'altres funcionalitats

Funcionalitat (Servidor)	Drupal BASE	+ mòduls estàndard	+ Mòduls de Cases rurals *	+ mòduls d'Activitats
Creació d'activitats		*		
Vista d'activitats portada				
Vista d'activitats + filtre				

Creació de camps de golf	
Creació de partides de golf	
Guardar estadístiques partides de golf	
Creació de la secció "El meu perfil"	**
Vista amb històric de partides de golf	
Vista d'activitats on participo/gestiono	
Apuntar-se a activats	
Creació interfície de serveis web per accés mòbil	

* la plataforma original de Cases rurals només té mínimes interseccions amb el mòdul d'activitats

** Aquesta funcionalitat contindrà contingut desenvolupat en el desenvolupament paral·lel. En aquest document s'explicarà només la realitzada en aquest projecte

Llegenda de la taula:

Funcionalitat no realitzada en aquest PFC

Funcionalitat parcialment realitzada

Funcionalitat realitzada

Les funcionalitats que ha de complir el client fet amb Android no apareixen al llistat ja que no disposem de cap versió prèvia o plataforma existent desenvolupada on poder comparar.

Funcionalitat (Client)
Login/Logout usuari
Sincronització de les dades

Visualització de partides
Actualització de partides existents
Creació de noves partides

1.2.2 Detall de les funcionalitats

En aquest punt s'explicaran breument les funcionalitats del sistema definides a l'apartat anterior. Les funcionalitats han estat agrupades segons la part del desenvolupament del projecte a la qual afecten.

Totes les funcionalitats quedaran detallades amb més profunditat ales seccions on s'indica l'anàlisi fet durant l'execució de les iteracions del projecte.

Portal web basat en Drupal:

1. **Gestió d'Activitats**

   - **Gestió d'activitats (alta/baixa/modificació):** Modificar la classe activitat existent per poder afegir més informació relativa a l'activitat i que aquesta quedi agrupada de forma lògica al ser visualitzada per pantalla.

   - **Vista d'activitats agrupades per tipus:** Vista que permet visualitzar les activitats agrupades en diferents categories amb diferents opcions de filtratge per obtenir les que més ens interessin.

   - **Permetre apuntar-se a les activitat:** Els usuaris podran apuntar-se a les activitats, tant creades per administradors, com activitats creades pels propis participants.

   - **Vista de les activitats on participo:** Vista que permet visualitzar les activitats a les quals m'he apuntat fent servir la funcionalitat anterior.

   - **Vista de les activitats que gestiono :** Vista que permet visualitzar les activitats que jo he creat com a usuari i saber, ràpidament, la quantitat d'usuaris que s'han apuntat.

2. **Gestió de partides de Golf**

   - **Gestió de partides de golf (alta/baixa/modificació):** Permetre als usuaris la creació de partides de golf així com disposar d'una visualització del resultat de la partida de la forma més amigable possible als jugadors de golf.

   - **Gestió de camps de golf (alta/baixa/modificació):** Permetre als administradors la creació de camps de golf on els usuaris podran crear partides. També s'ha de permetre la visualització de la informació del camps a tots els usuaris.

   - **Vista de les partides de l'usuari autenticat:** Vista on poder veure l'històric de partides jugades agrupades per camp de golf.

   - **Gestió d'estadístiques de les partides de golf:** Poder visualitzar de forma gràfica per a cada partida les estadístiques de l'usuari en relació al camp, al seu històric i a la resta de jugadors.

3. **Creació del perfil d'usuari:** En aquesta secció l'usuari podrà accedir a la informació que ha registrat en el portal (tant a nivell d'activitats i partides de golf com de cases rurals). En el nostre desenvolupament mostrarem les activitats que gestiono, les activitats on participo i l'històric de partides de golf. El projecte paral·lel a aquest desenvolupament s'encarregarà de mostrar la informació relacionada amb les cases rurals. S'haurà de posar en comú el resultat de cada desenvolupament per crear una secció conjunta.

4. **Exportació de serveis per ser explotats per aplicacions externes:** Crearem serveis que puguin ser accessibles remotament, en el nostre cas un dispositiu mòbil Android, per poder gestionar les partides de golf dels usuaris del portal basat en Drupal.

Aplicació per a dispositius mòbils Android:

1. **Login/Logout usuari**: El client haurà de ser capaç de crear una sessió al portal Drupal de l'usuari indicat

2. **Sincronització de les dades:** Permetre la sincronització de dades entre el portal Drupal i l'aplicació mòbil per obtenir les partides de l'usuari, els diferents camps de golf, i els possibles jugadors de cada partida.
3. **Visualització de partides:** Permetre veure el resultat d'una partida ja creada.
4. **Actualització de partides existents:** Permetre actualitzar la puntuació d'una partida de golf.
5. **Creació de noves partides:** Permetre la creació de noves partides de golf amb les mateixes opcions que tenim al portal Drupal.

1.3 Tecnologia utilitzada

Per la part del servidor web, la tecnologia utilitzada venia marcada per la decisió que es va prendre en la primera versió del desenvolupament, no obstant, es va investigar sobre diferents alternatives de codi obert (com poden ser Joomla o Liferay) per veure si aportaven algun avantatge. Aquestes alternatives van ser descartades degut a que el canvi que suposaven era pràcticament nul o contrari a la finalitat del sistema amb l'inconvenient d'haver de migrar un portal d'una plataforma a una altra, cosa que quedava fora de l'abast del projecte.

Per una altra banda, en la part client, hem decidit crear una aplicació per a dispositius Android, ja que ens és interessant conèixer el funcionament d'uns dels sistemes operatius mòbils amb major creixement, tant en número d'usuaris com en número de dispositius on el podem trobar.

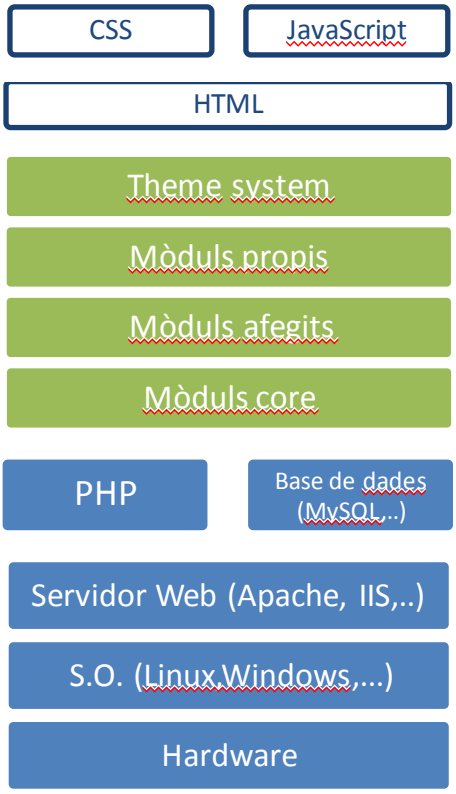
1.3.1 Drupal

Drupal és un sistema gestor de continguts modular que permet gestionar i organitzar una gran varietat de continguts per a un lloc webs.

Drupal es programari lliure, amb llicència GNU/GPL, escrit en PHP, que és desenvolupat i actualitzat per una comunitat molt activa d'usuaris. S'ha guanyat la reputació de ser un software de qualitat, que respecta els estàndards web, tot posant èmfasi en la usabilitat de l'aplicació així com la consistència de tot el sistema.

L'avantatge clau de Drupal és com planteja la seva modularitat, amb una alta granularitat per funcionalitats bàsiques i amb mòduls de mida considerable especialitats, aquesta dualitat fomenta que hi hagi un univers de mòduls (més de 2000) amb pocs solapaments]

Com hem explicat, el sistema es basa en codi PHP. A part d'un intèrpret de PHP, Drupal necessita un servidor web per a poder funcionar. En el següent gràfic podem observar l'arquitectura d'un projecte desenvolupat amb Drupal separat en diferents capes.



Il·lustració 3 - Torre de funcionament de Drupal

En blau i a més baix nivell ens trobem el que necessita el nostre servidor per poder fer funcionar Drupal. Bàsicament, és disposar en la màquina d'un servidor web, d'un intèrpret de PHP i un sistema de gestió de bases de dades relacionals.

En color verd trobem pròpiament el gestor de continguts Drupal. En les 2 capes inferiors trobem el que ens ofereix per defecte per un funcionament bàsic. En el següent nivell trobem els mòduls que hem anat afegint per obtenir les funcionalitats desitjades. Finalment, ens trobem amb el sistema que gestiona els diferents temes amb que es pot mostrar el contingut per generar els HTML, CSS i fitxers JavaScript necessaris per poder visualitzar el nostre sistema web.

1.3.2 Android

Android és un sistema operatiu obert , propietat de Google, basat en una versió modificada de Linux enfocat a dispositius mòbils.

A diferencia d'altres SO per a dispositius mòbils com iOS o Windows Phone, Android es desenvolupa de forma oberta i es pot accedir al codi font així com conèixer el llistat d'incidències o reportar-ne de noves.

Actualment, existeixen més de 200.000 aplicacions per a Android. Per a poder desenvolupar en Android, únicament cal disposar de coneixements de Java, llenguatge amb que s'escriuen les aplicacions, i disposar del kit de desenvolupament de software "SDK[3]" que Google proporciona de forma gratuïta.

Per poder disposar d'un entorn de desenvolupament per Android haurem de disposar dels següents elements:

[3] http://developer.android.com/sdk/index.html

2 Planificació

2.1 Metodologia

En la realització del projecte s'ha seguit una metodologia àgil per a la gestió de projectes basada en *Scrum*[4].

Aquesta metodologia segueix un paradigme iteratiu i incremental que consisteix en realitzar entregues parcials i regulars del producte final, sempre prioritzades per el benefici que aporten al receptor (normalment el client) del projecte a realitzar.

S'ha escollit Scrum com la metodologia a utilitzar per sobre d'altres, ja que ens aporta els següents beneficis:

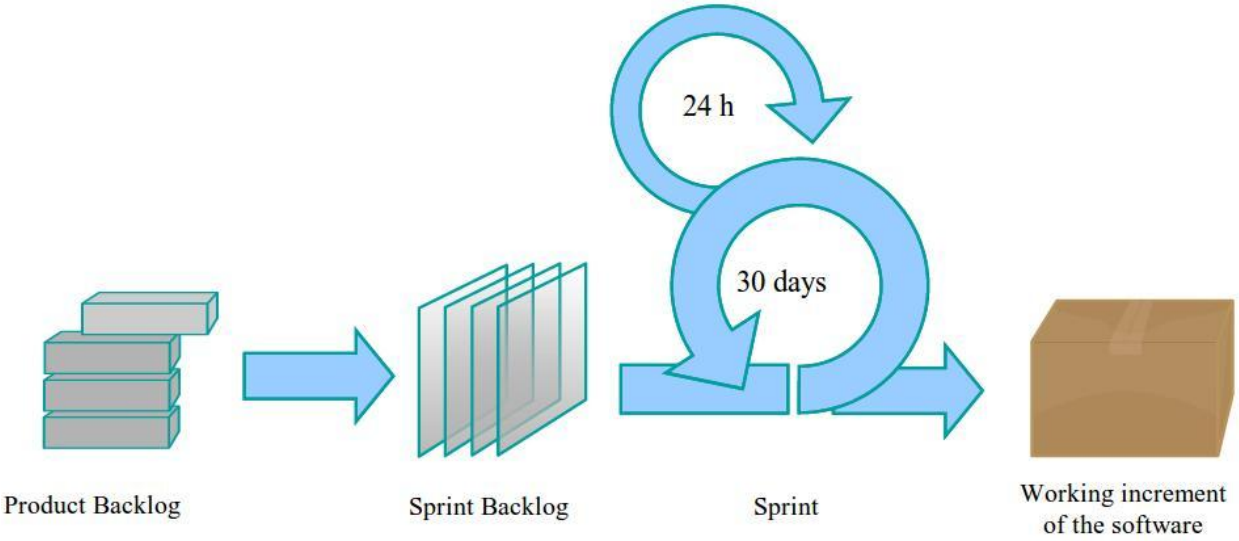
- Amb *Scrum* el risc de no completar el desenvolupament disminueix, ja que al finalitzar cada iteració disposem d'una part del software final operatiu.
- Les funcionalitats queden ordenades segons la seva importància i el seu cost a alt nivell. Tenint en compte els recursos disponibles es pot preveure, amb certa garantia, el conjunt de funcionalitats mínimes a entregar al client en un temps determinat.
- En cas d'arribar a l'última iteració i esgotar els recursos disponibles per al desenvolupament, el client disposarà de totes les funcionalitats de més valor realitzades en iteracions prèvies, mancant únicament les de menys valor per al producte final.
- Al finalitzar cada iteració el client a pot començar a utilitzar el sistema, de manera que baixa la seva incertesa sobre el producte final, alhora que descobreix els possibles canvis sense esperar a tenir el sistema acabat. No obstant, aquest procés pot provocar l'endarreriment del projecte si el client demana moltes modificacions al final de cada iteració.

[4] http://en.wikipedia.org/wiki/Scrum_(development)

2.1.1 Desenvolupament

Un projecte *Scrum* s'executa en blocs temporals curts i fixes (en aquest projecte s'han dut a terme 8 iteracions d'un mes natural). En cada iteració (o *sprint*) s'ha d'acabar un número de funcionalitats que poden ser entregades al client al final d'aquesta.

El següent gràfic presenta en forma de diagrama el procès iteratiu de realitzar un projecte amb metodologia *Scrum*:



Il·lustració 4 - Petit diagrama sobre com es porta un projecte basat en Scrum [5]

En primer terme ens trobem amb el *Product Backlog*. És un document que conté tots els requeriments indicats per el client en alt nivell, prioritzats segons el valor que el client els hi donaria un cop finalitzats. També contè una estimació de la dificultat i el número d'hores necessàries per a poder realitzar cada funcionalitat requerida.

En la primera iteració (Iteració 0) es planifiquen i distribueixen els objectius i abast del projecte assignats segons la seva prioritat en les iteracions. En la resta d'iteracions s'ha de prioritzar el benefici que pot aportar al client, el cost de desenvolupament i mitigar els possibles riscos tenint en compte que els objectius a realitzar en una iteració encaixin en la duració d'aquesta.

Acabada la Iteració 0, s'entregarà al client el *Product Roadmap*, document que conté el *Product Backlog* de totes les iteracions. Aquest document he de ser aprovat amb el client de forma conjunta.

[5] http://upload.wikimedia.org/wikipedia/commons/5/58/Scrum_process.svg

Un cop aprovat, es començaran a desenvolupar les iteracions del desenvolupament. Cada iteració tindrà com a objectiu completar un increment del producte que sigui demostrable (entregant al client un bloc de funcionalitat completa) i funcional. A més del desenvolupament, tota la documentació relacionada amb les funcionalitats fetes durant la iteració i la integració amb la feina feta a les iteracions anteriors seran finalitzades abans d'acabar el termini (el mes natural).

En tot projecte *Scrum* també es defineixen diversos rols, englobats en dos grans grups, els *porcs* (els que estan compromesos amb el projecte) i les *gallines*[6] (els que no formen part del projecte, però que s'han de tenir en compte).

En el grup del compromesos (porcs) trobarem al client (l'amo del producte), l'*ScrumMaster* (qui s'encarrega d'assegurar i guiar-nos perquè el projecte s'executi correctament) i l'equip de treball. Degut a que és un projecte a petita escala on només disposem d'una persona a l'equip de trebal, els dos últims rols seran realitzats per la mateixa persona.

Per altra banda, en el grup de les *gallines* només tindrem en compte als futurs usuaris de l'aplicació, ja que la resta de possibles rols (*Stakeholders* o *Managers* no tindrien pes en un projecte d'aquestes característiques. En aquest projecte, aquest rol serà simulat, donada la inexistència dels mateixos.

2.1.2 Requeriments funcionals

El sistema ha de conservar varis dels requisits que ja tenia la versió anterior:

- El sistema ha de poder ser accessible a usuaris externs al desenvolupament, mitjançant un servidor el qual ha de ser accessible per Internet.
- L'aplicació ha de ser totalment funcional, permetent als usuaris entrar al sistema i veure quines activitats hi ha.
- S'han de mantenir totes les propietats del sistema que no tenen a veure amb el desenvolupament realitzat en aquest projecte, bàsicament la gestió de cases rurals.
- S'ha de portar un registre de fitxers modificats i d'accions realitzades, per si el client volgués tornar a una versió anterior sense cap tipus de problema.

[6] Més informació sobre els rols del Scrum: http://www.dosideas.com/wiki/Roles_De_Scrum

- Tot nou desenvolupament ha d'encaixar en el tema (aparença) que hi ha actualment a la web, que és del gust del client. Si es presentes una nova proposta de disseny s'hauria de valorar l'esforç que s'hauria de realitzar per implantar un nou tema.
- Totes les dades s'han de desar en una base de dades MySQL.
- El portal serà desenvolupat en Drupal.
- El codi per a fer mòduls serà el llenguatge PHP, el donat per Drupal.
- Per poder anar omplint el resultat d'una partida de golf en el dispositiu mòbil no caldrà disposar de connexió amb el servidor en tot moment, únicament al final quan es vulguin desar les dades.

A més dels donats pel sistema a modificar, s'exigiran els següents:

- L'usuari ha de poder realitzar des del portal Drupal:
    - Apuntar-se a les activitats que vulguin realitzar
    - Crear noves activitats
    - Crear partides de golf
    - Consultar partides de golf existents i estadístiques
- L'usuari ha de poder consultar un històric de les activitats on ha participat així com de les activitats que ha creat/gestionat.
- L'usuari ha de disposar d'una secció on poder accedir a la seva informació de forma organitzada i clara.
- L'usuari ha de poder realitzar des de l'aplicació mòbil:
    - Entrar al sistema (login)
    - Crear partides de golf
    - Consultar partides de golf existents
- La informació relativa a les partides de golf i els camps de golf s'han de poder visualitzar d'una forma més clara i fàcil que la que permet Drupal per defecte.
- Aquest projecte serà desenvolupat paral·lelament a un altre que tracta sobre el mateix lloc web. Un cop acabats tots dos desenvolupaments, s'hauran d'ajuntar els dos desenvolupaments per fer-ne un de sòlid amb totes les característiques intactes.

2.2 Estimació d'esforç

Un cop revisades les sol·licituts del client, es va realitzar una taula amb una estimació temporal en hores de l'esforç que s'ha de dedicar a les funcionalitats demanades. Les hores d'esforç no només són calculades pel nombre d'elements que hi intervenen, sinó també per la dificultat que comporten després d'haver estudiat el Drupal i com funciona. En el cas de l'aplicació per dispositius Android també s'ha estudiat la complexitat que comporta cada funcionalitat després d'estudiar com funciona el SDK d'Android.

La taula amb les hores reals de desenvolupament està situada al finalitzar la memòria.

Portal Drupal:

Funcionalitat	Elements que intervenen	Estimació esforç de programació
Gestió d'activitats	7 mòduls 2 pantalles	30 hores
Vista d'activitats agrupades per tipus	3 mòduls 1 pantalles	25 hores
Apuntar-se a les activitats	1 mòduls Actualitzar pantalla activitat	30 hores
Vista de les activitats on participo	3 mòdul 1 pantalles	20 hores
Vista de les activitats que gestiono	3 mòduls 1 pantalla	20 hores
Gestió de partides de golf	7 mòduls 2 pantalles	30 hores
Gestió de camps de golf	4 mòduls 2 pantalles	30 hores
Templates per visualitzar activitats, partides de golf i camps de golf	1 mòdul 3 pantalles	30 hores
Vista de les partides de l'usuari	3 mòduls	25 hores

	2 pantalles	
Gestió d'estadístiques de les partides de golf	2 mòduls 1 pantalla	60 hores
Creació del perfil d'usuari	1 mòduls 1 pantalla	30 hores
Exportació de serveis per aplicacions externes	3 mòduls	60 hores
Unificació projectes paral·lels Drupal	----------	20 hores
TOTAL	----------	410 hores

Aplicació per a dispositius mòbils Android:

Funcionalitat	Estimació esforç de programació
Login/Logout usuari	35 hores
Sincronització de les dades	60 hores
Visualització de partides	50 hores
Actualització de partides existents	30 hores
Creació de noves partides	45 hores
TOTAL	220 hores

A part de les hores per realitzar les funcionalitats s'han de sumar les hores que dedicarem a l'estudi de la plataforma Drupal així com la posada en funcionament del sistema i l'estudi de la plataforma Android més la creació de l'esquelet de l'aplicació.

- Estudi del CMS Drupal més coneixement i posar en funcionament la plataforma existent: **100 hores**
- Estudi de la plataforma Android, més creació esquelet de l'aplicació i proves de l'aplicació en dispositius reals:

  80 hores

El total d'hores previst tant per realitzar les funcionalitats pel portal en Drupal com crear l'aplicació client en Android suma un total de:

	Hores
Estudi CMS Drupal	100 hores
Desenvolupaments Drupal	410 hores
Estudi plataforma Android	80 hores
Desenvolupament Android	220 hores
TOTAL	**810 hores**

Es van descartar algunes de les idees proposades per el client per falta de temps en el desenvolupament. Una de les idees interesants descartades vs ser la inclusió de publicitat geolocalitzada depenent la localització de la casa rural o activitat que s'estigui consultant en aquell moment o, en cas d'estar mirant un altre tipus de contingut sense localització, mostrar la publicitat relativa a la zona on està conectat l'usuari. Una altra proposta que ha quedat descartada en aquest desenvolupament per manca de temps seria la gestió de tornejos de golf. En qualsevol cas, les funcionalitats que si es realitzaran tindran en compte el fet de que aquestes funcionalitats es puguin adaptar al projecte en posteriors versions.

2.3 Recursos

Basant-nos en les funcionalitats a incloure, es va fer un llistat de recursos que s'utilitzaran durant o després del desenvolupament:

- 1 Cap de projecte, que en aquest cas farà de client.
- 1 Programador / Desenvolupador
- 1 Ordinador amb el hardware necessari per a poder executar les eines necessàries per al bon desenvolupament del projecte. A més, tindrà que tenir tot el programari necessari instal·lat i configurat.
- Un servidor amb connexió a Internet per poder accedir al portal en qualsevol moment.
- Diferents dispositius mòbils amb diferents versions del SO Android i diferents resolucions de pantalla.

2.4 Calendari d'activitats

Basant-se en les 2 taules de l'apartat 2.2 s'ha intentat repartir les funcionalitats a implementar per tenir una càrrega equilibrada de les iteracions. No obstant, s'ha tingut en compte que es disposa únicament d'un desenvolupador que no pot dedicar el mateix temps a cada iteració. Per tant, les funcionalitats han quedat repartides basant-nos en més factors que únicament la càrrega de treball.

A la taula següent podrem veure com s'han distribuït les funcionalitats repartides en les setmanes del mes que ocupen (cada iteració serà d'un mes de durada). Noteu que per problemes de calendari del desenvolupador en el més de maig i octubre no ha pogut pràcticament dedicar-se al projecte degut a altres compromisos. Per això les iteracions 3 i 7 tenen 2 mesos de durada tot i tenir una càrrega semblant de treball. Destacar també que en alguns mesos d'estiu (especialment juliol) s'ha pogut dedicar menys hores degut a les vacances del desenvolupador.

<table>
<tr><th>Sem.</th><th>Febrer
Iteració 0</th><th>Març
Iteració 1</th><th>Abril
Iteració 2</th><th>Maig/Juny
Iteració 3</th><th>Juliol
Iteració 4</th><th>Agost
Iteració 5</th><th>Septembre
Iteració 6</th><th>Oct./Nov.
Iteració 7</th><th>Decembre
Iteració 8</th></tr>
<tr><td>1</td><td rowspan="2">Estudiar plataforma Drupal. Instal·lació del software necessari</td><td rowspan="2">Gestió d'activitats</td><td>Gestió de partides de golf</td><td>Apuntar-se a les activitats</td><td rowspan="2">Vista de les partides de l'usuari</td><td>Unificació projectes paral·lels Drupal</td><td rowspan="2">Estudi/ins-tal·lació plataforma Android</td><td rowspan="2">Sincronitza-ció de les dades</td><td rowspan="2">Creació de noves partides</td></tr>
<tr><td>2</td><td>Vista de les activitats on participo</td><td rowspan="3">Gestió d'estadísti-ques de les partides de golf</td><td rowspan="3">Exportació de serveis per a aplicacions externes</td></tr>
<tr><td>3</td><td>Estudii instal·lació de la plataforma existent.</td><td>Vista d'activitats agrupades per tipus</td><td>Vista de les activitats que gestiono</td><td rowspan="2">Creació del perfil d'usuari</td><td>Creació esquelet aplicació android</td><td rowspan="2">Visualització de partides</td><td>Actualització de partides existents</td></tr>
<tr><td>4</td><td>Planificació del projecte</td><td>Gestió de camps de golf</td><td>Templates per acivitats, partides i camps de golf</td><td>Login usuari</td><td>Testing aplicació en dispositius reals</td></tr>
<tr><td>Total</td><td>100 hores</td><td>85 hores</td><td>100 hores</td><td>90 hores</td><td>55 hores</td><td>80 hores</td><td>90 hores</td><td>110 hores</td><td>100 hores</td></tr>
</table>

2.5 Riscos

Durant el desenvolupament de qualsevol projecte, poden aparèixer diferents problemes que s'han d'afrontar i resoldre per assolir els objectius del projecte. Si abans de trobar-nos en una situació no desitjada som capaços preveure aquests riscos, podrem solucionar-los de forma més ràpida i eficient. Alguns dels riscos i les seves descripcions, estratègies de mitigació i accions de contingència van expressades a continuació:

- Endarreriment al finalitzar una iteració
    - **Descripció:** No acabar una iteració en el temps previst, amb risc de enrederir tot el projecte.
    - **Impacte:** Mig
    - **Probabilitat:** Mitja
    - **Accions mitigació:** Seguir el calendari d'activitats programat i mantenir de forma freqüents reunions pel seguiment del projecte
    - **Accions de contingència:** En cas que sigui possible utilitzar hores de desenvolupament d'una iteració posterior en la que s'hagi previst que és factible reduir algunes hores de la planificació prevista inicialment.
- No poder assolir una funcionalitat desitjada pel client
    - **Descripció:** S'arriba a un punt en que el desenvolupador no trobar cap forma de satisfer una necessitat en un temps que no posi en perill la data màxima permesa per finalitzar el projecte
    - **Impacte:** Alt
    - **Probabilitat:** Baixa
    - **Accions de mitigació:** Estudiar en la iteració 0 la viabilitat en temps de cada funcionalitat avisant al client en possibles modificacions del calendari.
    - **Accions de contingència:** Es pararà el desenvolupament de la funcionalitat i es continuarà amb la següent que no depengui d'aquesta. Es descomptarà del pressupost les hores assignades a aquesta funcionalitat.

- La finalització del projecte té un retràs de més de 15 dies
    - **Descripció:** S'arriba a la finalització del projecte i no s'ha acabat. El temps previst de més es superior a les dues setmanes.
    - **Impacte:** Alt
    - **Probabilitat:** Baixa
    - **Accions de mitigació:** Es portarà al dia la taula del calendari d'activitats
    - **Accions de contingència:** S'acabarà el més ràpid possible, intentant no demorar gaire la presentació així com refer el pressupost inicial.
- No disposar de dispositius mòbils amb SO Android
    - **Descripció:** S'arriba al procés de proves de l'aplicació Android i no es disposa de terminals reals per realitzar proves.
    - **Impacte:** Alt
    - **Probabilitat:** Mitja
    - **Accions de mitigació:** El desenvolupador renovarà el seu terminal per un amb Android durant realització de l'aplicació.
    - **Accions de contingència:** El codirector del projecte disposa actualment d'un mòbil amb SO Android. També es faran proves mitjançant un emulador de dispositiu mòbil facilitat per l'entorn de desenvolupament utilitzant diferents resolucions i versions de Android.

2.6 Pressupost

D'acord amb el càlcul d'hores anterior, el programador júnior tindrà un cost aproximat de:

- 810 hores de desenvolupament a 20 euros l'hora del programador = **16.200 €**

S'ha decidit realitzar reunions de seguiment a partir de la iteració 1. Es faran 2 reunions de seguiment per iteració de durada d'una hora. En cadascuna d'elles assistirà el cap de projecte i els representants que el client consideri necessaris. El cost per hora del cap de projecte duplica el cost per hora del programador. Per tant:

- 16 reunions d'una hora a 40 euros l'hora = **640 €**

La figura de cap de projecte existeix en aquest projecte, però actua de client així que no tindrà cap cost.

En el preu no ve inclòs l'opció de hosting. Després d'estudiar diferents ofertes i solucions podem recomanar la solució de hosting professional de l'empresa dinahosting[7] (disposa del hardware necessari per poder allotjar el portal) a un preu anual a partir de **141,13€** (IVA inclòs). En cas de no disposar de domini s'haurà de sumar un preu de **14€** (també anual).

En cas de voler contractar el servei de hosting aquest import només s'haurà de començar a pagar l'últim mes de desenvolupament, ja que al ser un recurs ALAP (As Last As Possible) no es contractarà abans de cara a estalviar diners.

Tot i que el desenvolupador ja disposa d'una màquina on desenvolupar el projecte es cobrarà al client el cost equivalent de la màquina pel temps que durà el projecte partint que el temps d'amortització de la màquina es de 2 anys (temps on es tornarà a renovar l'equip). El projecte té una durada aproximada d'11 mesos i el cost de la màquina es de 1100€. Per tant cost de l'equip en el projecte és de **504,2 €**.

També es necessitarà software d'edició per a desenvolupar i mantenir la web. Tot el software escollit és gratuït, així que el cost en software serà de **0€.**

Total del projecte: (sense comptar hosting) :

RECURS	COST
Programador	16.200 €
Cap de projecte	640 €
Hardware	504,20€
	17.344,20€

[7] Més informació de la solució de hosting a: https://dinahosting.com/es/hosting/hosting-profesional

2.7 Metodologia de treball en equip

Paral·lelament a aquest projecte s'estaven desenvolupant funcionalitats en un altre projecte de similars característiques (relacionades amb la millora de la gestió de cases rurals). Les funcionalitats de l'altre projecte són totalment compatibles i independents del desenvolupament realitzat en aquest projecte. Únicament interaccionen en el punt del perfil d'usuari on cada projecte part de la informació a mostrar, un relacionat amb les cases i el que ocupa aquest document relacionat amb les activitats.

Donat que el client volia que s'integessin els dos projectes en un de sol es van planificar diferents dates on s'anirien ajuntant les funcionalitats fetes en cada desenvolupament, no deixant tota la migració per al final.

Aquestes migracions s'han de realitzar amb molta cura, fent imprescindible una detallada documentació dels fitxers i configuracions que aportava cada projecte al sistema, ja que en cas d'error seria dificultós localitzar-lo i reparar-lo.

Tenint en compte que els dos projectes finalitzaven en periodes similars (com a mínim el desenvolupament fet per Drupal) es varen planificar diferents reunions entre el client i els desenvolupadors dels projectes. El més important que s'ha fet en aquestes reunions ha sigut posar en comú una llista de tasques a fer per a la bona migració dels sistemes.

Finalment, la migració entre les màquines es va fer realitzar sense gaires dificultats i dintre dels plaços determinats, sense haver de retrasar cap dels dos desenvolupaments.

En totes les reunions es va parlar del desenvolupament en general, intentant que la feina d'un i d'altre s'assembles el màxim possible, fent així un codi uniforme tot i que el fet de programar en Drupal ja fa que el codi dels mòduls sigui força uniforme.

3 Iteració 0

3.1 Llista d'activitats de la iteració

1- Planificar i distribuir els objectius i abast (descrits en l'apartat 2 per a facilitar la comprensió del desenvolupament)
2- Identificar objectius i requeriments d'alt nivell. També identificar els requisits no funcionals del sistema.
3- Preparar tot l'entorn de desenvolupant, incloent disposar del hardware adequat així com tenir tot el software necessari instal·lat.
4- Revisar l'arquitectura del sistema.
5- Estudiar el funcionament general del Drupal i el del sistema existent en concret.
6- Idear la estructura bàsica en el disseny del sistema web.

3.2 Requisits no funcionals

En aquest punt seran exposats els requisits no funcionals de l'aplicació. Són aquells que tenen relació amb les característiques del sistema en general, no només a alguna funcionalitat especifica. Els trobem en el següent llistat:

- La plataforma estarà integrament en anglès. Drupal permet la creació de sites en diferents idiomes. El fet de mantenir i crear contingut en diferents idiomes suposa un esforç extra en el temps que faria reduir la llista de funcionalitats.
- El sistema ha de ser capaç de donar resposta als usuaris en un temps acceptable i uniforme tenint en compte les possibilitats tecnològiques de les eines emprades.
- Complir en tot moment totes les normes referents a la confidencialitat dels clients, així si un usuari demana que les seves dades siguin esborrades es procedirà a executar la seva petició.
- Garantir que el servidor haurà d'estar el 99% del temps online, per assegurar el bon funcionament de l'eina.

- El sistema ha d'estar escrit de manera escalar, de manera que noves funcionalitats i requeriments relacionats puguin ser incorporats afectant el codi de la menor manera possible. S'ha de tenir en compte que el desenvolupat en aquest projecte pot ser la base d'un altre projecte que amplii les funcionalitats del sistema.
- El sistema ha de permetre que qualsevol usuari (sigui expert o no) el pugui utilitzar sense complicacions. Per tant, tindrem en la facilitat d'ús una de les prioritats a l'hora de prendre decisions de disseny.
- La informació serà verificada prèviament pel sistema abans de ser desada a la base de dades. En cas d'error, l'usuari rebrà un missatge informatiu de l'error facilitant la solució d'aquest.
- L'aplicació mòbil per a dispositius Android ha de ser compatible amb tots els dispositius amb versions del SO a partir de la 1.5.
- L'aplicació mòbil s'haurà de visualitzar correctament independentment de la resolució del dispositiu.
- El sistema ha de ser fàcil d'instal·lar en totes les plataformes de hardware existents, sempre que compleixin uns requisits mínims.
- El sistema quedarà documentat per facilitar el manteniment de sistema o la creació de noves funcionalitats per a futurs desenvolupadors.
- L'administrador de la plataforma ha de comptar amb una interfície gràfica d'administració fàcil d'utilitzar.
- L'accés al sistema d'administració de la plataforma només s'atorgarà a persones físiques i ha de ser donat per l'administrador total del sistema.

3.3 Disseny conceptual de l'aplicació

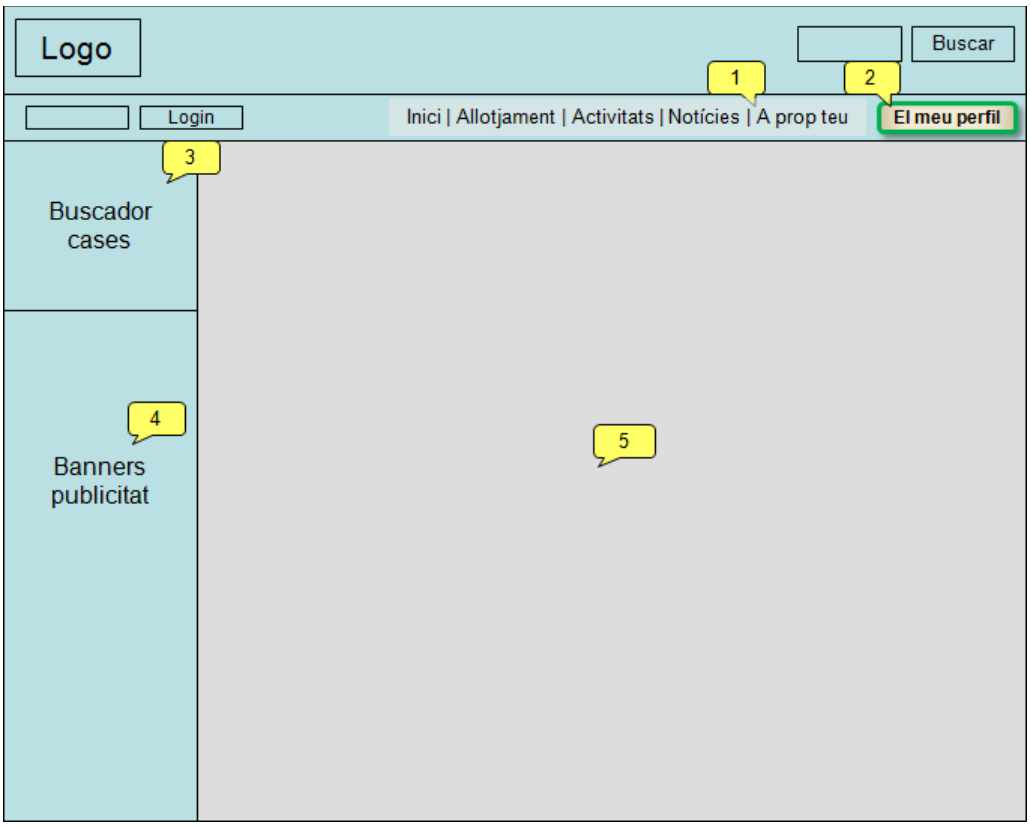
En aquest apartat s'intentarà explicar com els usuaris navegaran pel sistema. S'explicarà la navegació entre les diferents pantalles del sistema així com el disseny que presentaran les pantalles, que seguiran un patró únic.

3.3.1 Disseny gràfic

Ens centrarem primer en explicar la distribució de la pantalla sense fixar-nos en cap funcionalitat en si, el que seria el marc de l'aplicació.

Al partir d'un marc base, de la versió prèvia, es va decidir no modificar gaire l'estructura general ja que aquesta distribució ja era de gust del client.

A continuació podem veure com quedaria la distribució de la pantalla:



Il·lustració 5 - Esquema bàsic de la pàgina

Seguint els números de l'esquema trobem els blocs que conformen el marc de l'aplicació:

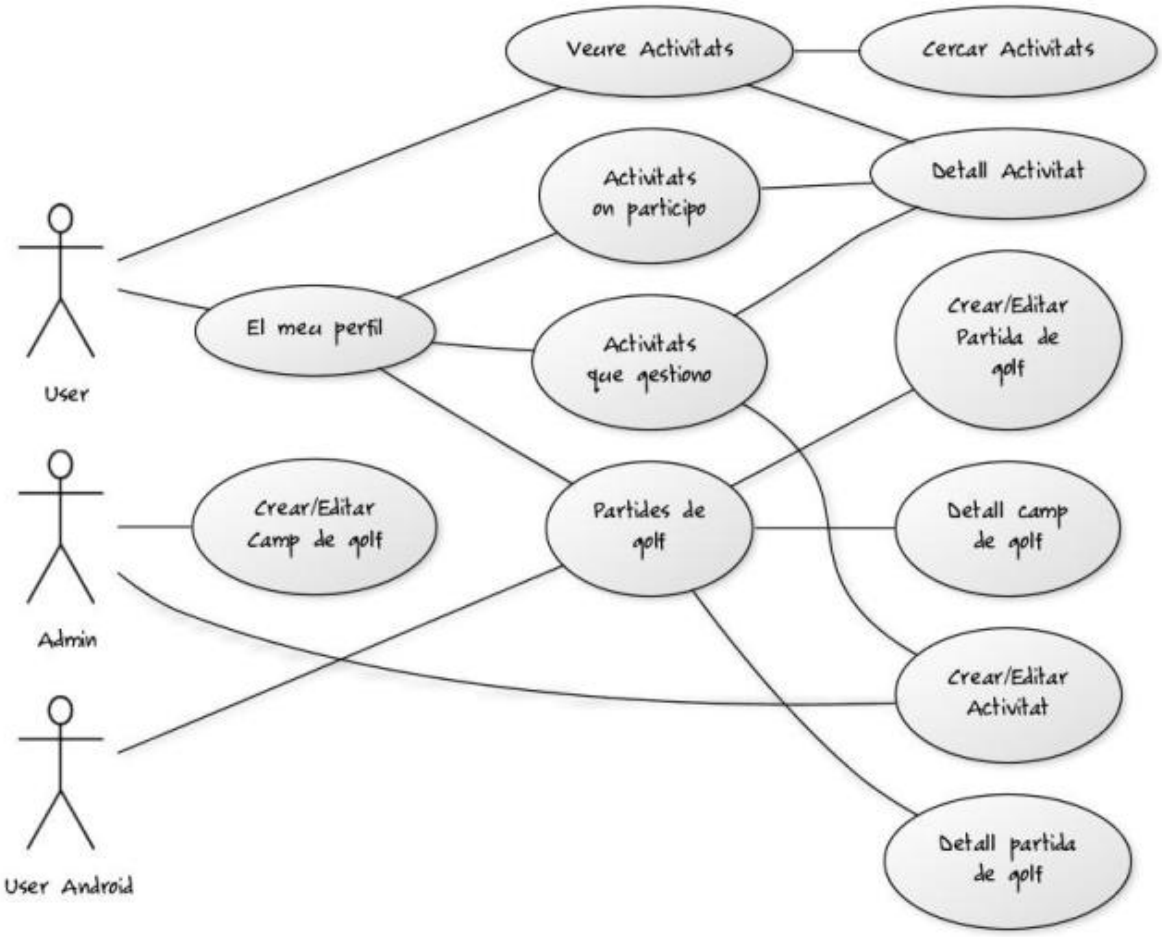
1- Menú de navegació entre les diferents seccions del portal
2- Botó que portarà a la secció "El meu perfil". Aquest botó ha de ser visible clarament, tot diferenciant-se del menú de navegació.
3- A l'esquerra trobarem el cercador de cases rurals.
4- Sota el cercador s'ubicaran els banners publicitaris.

5- En aquest bloc anirà el contingut de la pàgina en que ens trobem.

El disseny gràfic de l'aplicació mòbil quedarà definit en la iteració 6 quan es defineixi l'esquelet de l'aplicació.

3.3.2 Mapa de navegació

En el mapa de navegació que veurem a continuació detallarem com un usuari pot navegar entre les pantalles que afectaran al nostre desenvolupament (gestió d'activitats). El salt entre funcionalitats podrà ser més directe que el que mostra el mapa degut al menú de navegació de que disposem. Destacar també la inclusió de l'usuari que accedirà al sistema a través de l'aplicació mòbil, on només tindrà accés a les partides de golf.



Il·lustració 6 - Mapa de navegació corresponent a la plataforma Drupal-Android

3.4 Arquitectura

En aquest apartat s'expandirà el gràfic situat a l'apartat 1.3.1 sobre l'arquitectura necessària per fer funcionar un portal en Drupal, detallant en cada nivell les característiques necessàries per al bon funcionament de la plataforma desenvolupada.

- **Hardware:** Es refereix a la màquina física on estarà implementat el desenvolupament, no cal que sigui una màquina molt potent, únicament ha de ser capaç de processar totes les consultes demanades en els requisits.
- **Sistema Operatiu:** No cal que sigui l'última versió de cap sistema, únicament ser capaç de poder contenir tot el software necessari (servidor web, intèrpret php, gestor de base de dades,...).
- **Servidor Web:** Utilitzarem Apache com a servidor web, ja que és un dels servidors més estesos i amb més documentació que podem trobar. A part, existeixen diferents paquets com LAMP (Linux) o WAMP (Windows) entre d'altres que en un únic paquet instal·len el servidor Apache, el intèrpret de PHP i el gestor de bases de dades MySQL. D'aquesta manera amb una única instal·lació disposem de tot el software necessari per tenir online un portal web basta en el gestor de continguts Drupal.
- **Intèrpret de PHP i Gestor de bases de dades:** En aquests 2 casos hem utilitzat les opcions integrades en la infraestructura proporcionada pel paquet de software WAMP.
- **Drupal (mòduls sistema):** En tota instal·lació de Drupal, per defecte s'instal·len uns mòduls que permeten el funcionament més bàsic del sistema. Aquest mòduls són anomenats els mòduls core ja que són el nucli del sistema. Com a mòduls core més importants podem trobar el mòdul Node (que permet la creació de continguts) i el mòdul User (que permet la creació d'usuari) entre d'altres.
- **Drupal (mòduls instal·lats):** A part dels mòduls core, en Drupal podem instal·lar una gran varietat de mòduls desenvolupats per la comunitat de desenvolupadors de Drupal que ens permeten afegir funcionalitats als sistema. Aquests mòduls els podem

descarregar i afegir al nostre sistema des de la pàgina oficial de Drupal[8]. De la mateixa manera que amb els mòduls core, totes les funcions que siguin instal·lades amb aquests mòduls seran accessibles per la resta del sistema, així que és habitual que un mòdul utilitzi funcions d'altres creant-ne dependencies entre mòduls.

- **Mòduls propis:** Si no es troba cap mòdul que satisfaci les necessitats del nostre sistema sempre podem crear-ne nosaltres un a mida, que fins i tot, el podem publicar, en cas de considerar que pot ser d'ajuda en altres projectes, tot i que normalment, aquests mòduls només tenen sentit per al sistema on són desenvolupats.

3.5 Elicitació de requeriments (Product Backlog)

Com hem vist en punts anteriors, el *Product Backlog* en *Scrum* representa la visió i expectatives del client respecte els objectius i entregues del projecte. Aquest document ajuda a saber la prioritat de cada funcionalitat pel client, així com la estimació inicial de hores i l'estimació després d'haver estudiar com s'han d'implementar (però no a posteriori). El valor de cada funcionalitat el dóna el client, després d'expressar les seves necessitats:

Funcionalitat	Prioritat pel client (0 min-10màx)	Estimació inicial	Estimació ajustada
Gestió d'activitats	8	30 hores	25 hores
Vista d'activitats agrupades per tipus	8	25 hores	25 hores
Gestió de camps de golf	4	30 hores	25 hores
Iteració 1	20	85 hores	75 hores
Gestió de partides de golf	8	30 hores	25 hores
Vista de les activitats on participo	4	20 hores	20 hores

[8] http://drupal.org/project/modules

Vista de les activitats que gestiono	4	20 hores	20 hores
Utilitzar templates per activitats, partides i camps de golf	6	30 hores	35 hores
Iteració 2	22	100 hores	100 hores
Apuntar-se a les activitats	7	30 hores	30 hores
Gestió d'estadístiques de les partides de golf	7	60 hores	65 hores
Iteració 3	14	90 hores	95 hores
Vista de les partides de l'usuari	7	25 hores	20 hores
Creació del perfil d'usuari	8	30 hores	30 hores
Iteració 4	15	55 hores	50 hores
Unificació projectes paral·lels Drupal	7	20 hores	25 hores
Exportació de serveis per aplicacions externes	7	60 hores	60 hores
Iteració 5	14	80 hores	85 hores
Creació esquelet aplicació (Android)	3	55 hores	65 hores
Login usuari (Android)	4	35 hores	35 hores
Iteració 6	7	90 hores	100 hores
Sincronització de les dades (Android)	5	60 hores	60 hores
Visualització de partides (Android)	8	50 hores	50 hores

Iteració 7	13	110 hores	110 hores
Creació de noves partides (Android)	7	45 hores	40 hores
Actualització de partides existents (Android)	7	30 hores	25 hores
Iteració 8	14	75 hores	65 hores

Tot i que idealment, en un projecte realitzat en Scrum les funcionalitats amb més prioritat pell client haurien d'anar en les primeres iteracions, podem veure que en aquesta taula no es segueix exactament aquest ordre. Això es degut a que només es disposa d'un únic desenvolupador i realitzar alhora funcionalitats en el Drupal i funcionalitats per a l'aplicació mòbil hagués endarrerit el final del projecte ja que s'hauría d'haver de formar-se simultàneament en les plataformes Drupal i Android. A més a més al tenir que estar canviant d'un entorn a un altre, baixaria la productivitat del desenvolupador.

No obstant, s'ha intentat que dintre de cada desenvolupament es realitzin primer les funcionalitats amb més valor pel client, sempre tenint en compte les possibles dependencies.

4 Iteració 1

4.1 Llista d'activitats de la iteració

4.1.1 Gestió d'activitats

1- Estudiar la creació/modificació actual d'activitats que tenim al sistema.

2- Adaptar el tipus de contingut actual "Activitat" a les necessitats del client permetent:

   a. Tipologia d'activitats

   b. Organització de la informació en blocs que facilitin la lectura de la informació

   c. Permetre camps exclusius segons la tipologia de l'activitat

4.1.2 Vista d'activitats agrupades per tipus

1- Estudiar el funcionament del mòdul Views de Drupal per a poder crear vistes de continguts

2- Investigar diferents opcions de visualització del resultat d'una vista.

3- Crear una vista de les activitats que permeti:

   a. L'agrupació d'activitats per tipus ordenades per data

   b. Permetre aplicar filtres de cerca per tipus, regió de l'activitat i data d'aquesta

4- Mostrar el resultat en forma de graella que faciliti la lectura i comprensió de la informació

4.1.3 Gestió de camps de golf

1- Estudiar la informació que necessitem guardar d'un camp de golf.

2- Crear un nou tipus de contingut per guardar informació sobre els camps de golf

3- Permetre relacionar les activitats de tipus "Golf" a un camp de golf que s'hagi inserit prèviament.

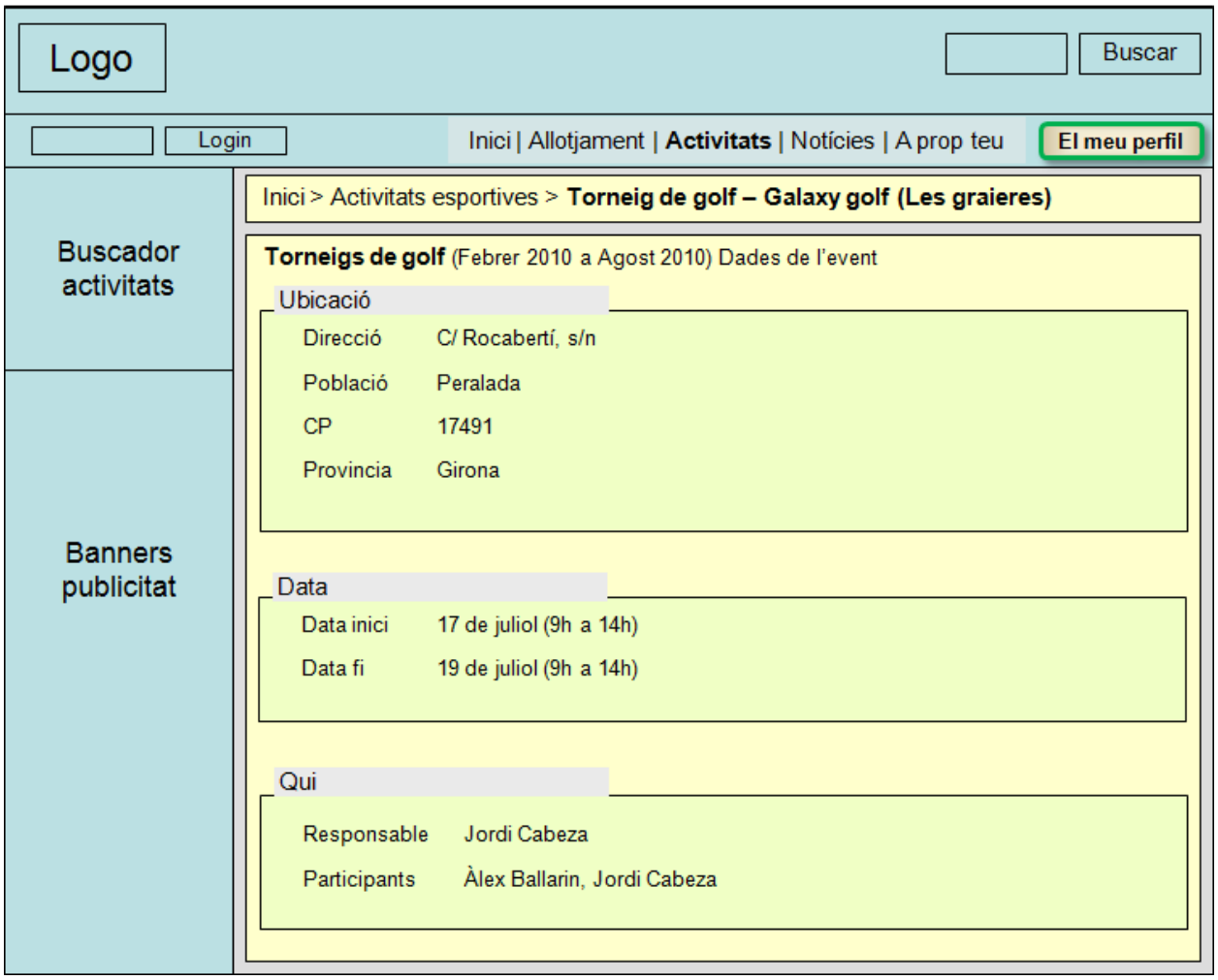
4.2 Anàlisi i disseny de les funcionalitats

4.2.1 Gestió d'activitats

Aquesta funcionalitat ja existia en la versió preliminar i, per tant, es podien crear activitats en el sistema. No obstant, el client desitjava ampliar la informació a mostrar d'una activitat, poder realitzar una classificació per tipus, i agrupar la informació de forma lògica en grups tenint en compte el lloc de l'activitat, la data, qui ho organitza i qui hi participa. Tots aquests aspectes no es tenien en compte en l'actual versió.

D'aquesta manera es va decidir reconstruir el tipus de contingut "Activitat" basant-nos en les directrius del client.

4.2.1.1Maqueta de la funcionalitat



Il·lustració 7 - Disseny preliminar de la funcionalitat

4.2.1.2Especificació detallada

Seguint les directrius del client i basant-nos en el disseny preliminar de la maqueta (es una maquest no completa ja que falten grups d'informació) el tipus de contingut "Activitat" quedarà definit de la següent forma:

Activitat:

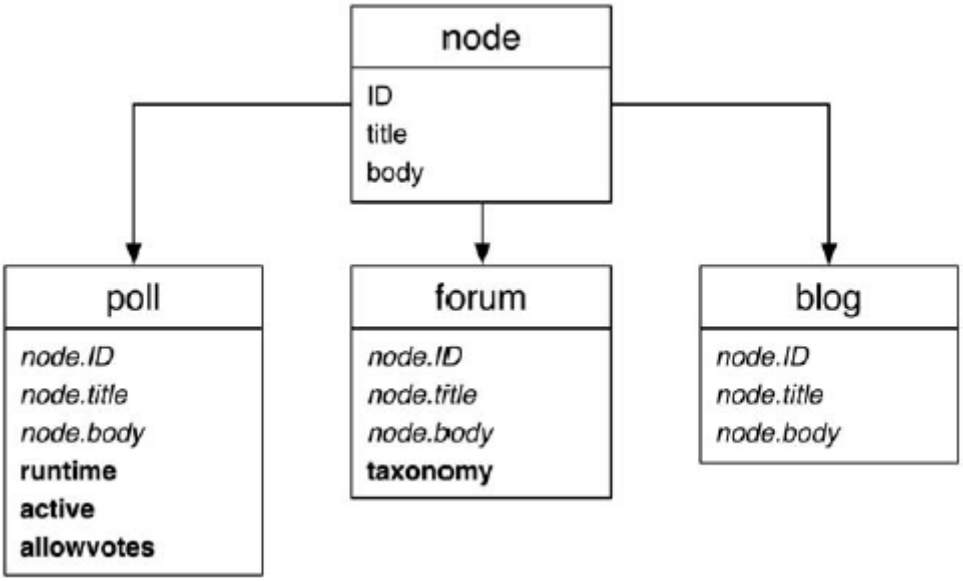
- **Títol:** Títol de l'activitat que servirà per identificar-la respecte a la altres de forma visual.

- **Tipus d'activitat:** Les activitats es podran classificar segons el tipus (Partides de golf, excursions,...). El llistat de tipus ha de ser ampliable i editable de forma fàcil per als administradors del sistema.
- **Ubicació:**
    - **Direcció:** Indicarà la direcció on es realitza l'activitat
    - **Població:** Població de l'activitat
    - **CP:** Codi postal de la direcció
    - **Província:** Província a la que pertany la població indicada
- **Golf (només si es de tipus golf)**
    - **Camp de golf:** Indica el camp de golf on es realitzarà l'activitat
- **Durada:**
    - **Data d'inici:** Indicarà la data d'inici de l'activitat
    - **Data fi:** Indicarà quan termina l'activitat
- **Qui:**
    - **Responsables:** Indicarà els usuaris que es fan responsables d'organitzar l'activitat
    - **Participants:** Usuaris que tenen intenció de participar en l'activitat
- **Descripció:**
    - **Descripció:** Text amb informació detallada de l'activitat
    - **Enllaç:** Enllaç a alguna url externa o interna al portal en cas que es consideri necessari per ampliar la informació sobre l'activitat
- **Galeria de fotografies:** L'usuari podrà pujar fotografies relatives a l'activitat

4.2.1.3Disseny de la funcionalitat

Drupal permet als usuaris la creació de continguts. A Drupal tots els continguts són considerats com a nodes. Llavors un node és una unitat de contingut en Drupal que conté la següent informació básica: identificador numeric, títol i descripció. Nosaltres com a desenvolupadors podem crear tipus de continguts que afegeixen informació a la que ens proporciona un node. D'questa manera un tipus de contingut es una subclase que deriva de la clase node i que amplia

els seus atributs. Ho podem veure clarament en el següent gràfic amb alguns tipus de contingut que dóna Drupal per defecte:



Il·lustració 8 - Pro Drupal Development 2nd Edition - Explicació del concepte tipus de contingut

En el nostre cas, per poder crear activitats, crearem el tipus de contingut "Activitat" que afegirà a la informació bàsica d'un node els camps que hem definit en el punt anterior.

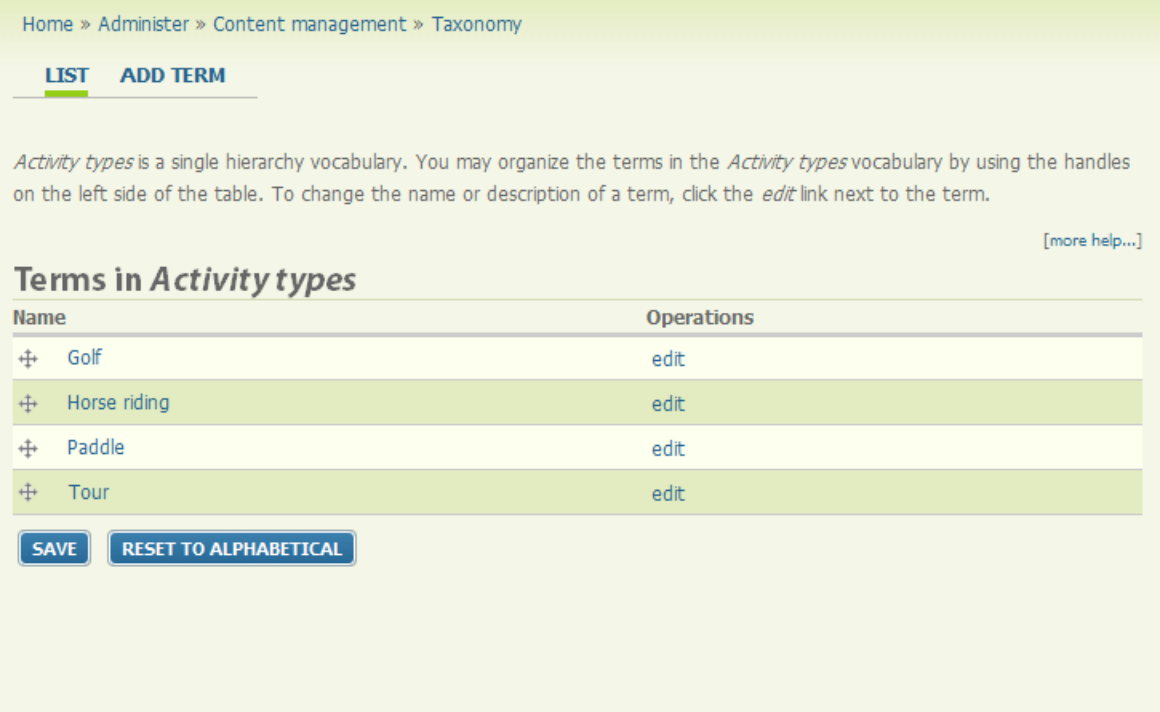
Per crear un tipus de contingut en Drupal disposem del mòdul Content Construction Kit (CCK)[9] que ens permet afegir nous camps als nodes creant noves clases o tipus de continguts. Un cop afegits els camps el propi Drupal s'encarrega de crear el formulari de creació/edició del nou tipus de contingut amb els atributs indicats. El propi Drupal realitza les validacions que nosaltres indiquem al afegir atributs (si són obligatoris o no, longitud,etc).

Després d'investigar el funcionament del mòdul CCK s'ha vist que era insuficient per poder satisfer totes les necessitats i s'han tingut que utilitzar altres mòduls que afegeixen funcionalitats a la creació de continguts. Per agrupar els camps es pot utilitzar el mòdul FieldGroup (inclòs en el CCK) que crea grups de camps per organitzar la informació de forma més coherent. A continuació farem un llistat dels camps que s'han afegit al tipus de contingut

[9] Més informació sobre el mòdul CCK a: http://drupal.org/project/cck

"Activitat" indicant la seva configuració i els mòduls utilitzats en cas que fossin necessaris agrupats dins del seu grup.

- **Tipus d'activitat:** Per afegir el camp tipus d'activitat el que primer s'ha creat és una taxonomia amb el mòdul Taxonomy[10]. El que ens interessa és classificar continguts. Essencialment una taxonomia en Drupal consisteix en un "vocabulari" de "termes". En el nostre cas crearem un vocabulari de tipus d'activitats on cada tipus serà un terme.

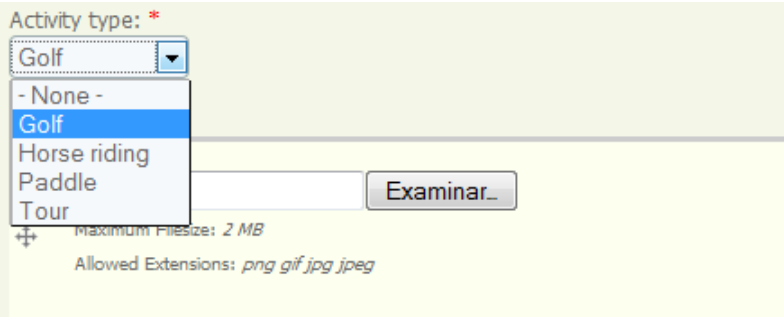


Il·lustració 9 - Vocabulari amb el tipus d'activitat

Amb la taxonomia creada, s'ha d'afegir-la al tipus de contingut. Els avantatges d'utilitzar una taxonomia són que ens permet utilitzar aquest mateix vocabulari en altres continguts i permetre la cerca de contingut així com facilitar el manteniment dels llistats, ja que només es troben en un lloc comú. Per indicar que un tipus de contingut està relacionat amb una taxonomia no cal crear un nou camp, només afegir la taxonomia creada. Però si ho fem d'aquesta manera no podrem controlar que el camp que indica el camp de golf de l'activitat només aparegui quan l'activitat es de tipus golf. Per poder controlar aquest esdeveniment afegirem la taxonomia al tipus de contingut

[10] Més informació sobre taxonomies a: http://drupal.org/node/774892

mitjançant el mòdul Content Taxonomy[11]. Aquest mòdul ens permet afegir una taxonomia com un atribut més del tipus de contingut. En el nostre cas, crearem l'atribut tipus d'activitat especificant que utilitzi el vocabulari que acabem de crear i que ens mostri els termes en una llista desplegable al haver poques opcions. El que ens trobarem en el formulari de creació quan vulguem crear una activitat serà el següent camp:

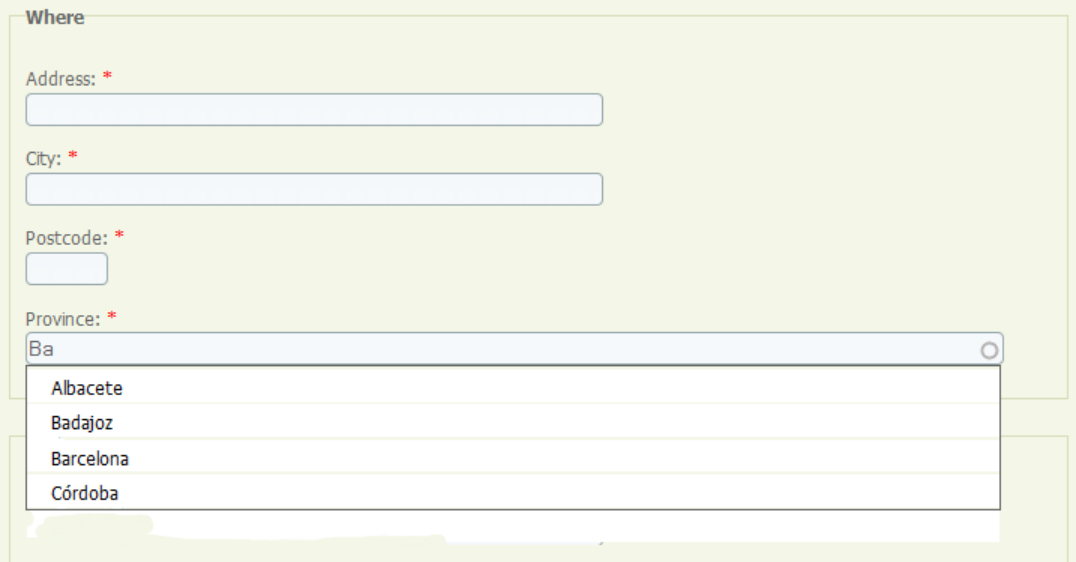


Il·lustració 10 - Atribut tipus d'Activitat

- **Ubicació:** Es crearà el grup d'atributs "group_where" que inclou els atributs que fan relació a la ubicació de l'activitat.
    - **Direcció:** S'afegirà un nou atribut obligatori de tipus "Text" d'una longitud de 60 caràcters.
    - **Població:** S'afegirà un nou atribut obligatori de tipus "Text" d'una longitud de 60 caràcters.
    - **CP:** S'afegirà un nou atribut obligatori de tipus "Text" d'una longitud de 5 caràcters.
    - **Província:** S'afegirà aquest atribut de la mateixa forma que el tipus d'activitat aprofitant-nos que el la versió prèvia del sistema ja s'havia inclòs una taxonomia amb les províncies que s'utilitza per les cases rurals. La diferencia d'aquest camp es que escollirem que es mostri utilitzant el widget Autocomplete (ve inclòs en el mòdul Content Taxonomy). D'aquesta manera se'ns mostrarà les opcions del camp conforme anem escrivint mostrant a l'usuari els termes del vocabulari que

[11] Més informació del mòdul Content Taxonomy a: http://drupal.org/project/content_taxonomy

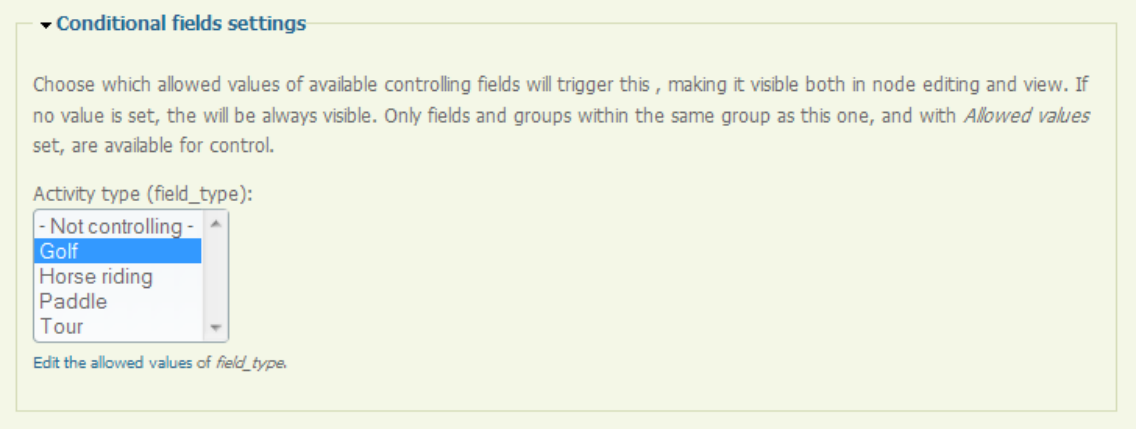
coincideixin amb el que s'ha escrit. S'ha escollit aquesta opció ja que utilitzar un llistat amb totes les províncies queda massa extens i dificulta l'elecció de la província. El resultat és el següent:



Il·lustració 11 - Atribut província amb widget autocomplete

- **Golf:** Es crearà un nou grup d'atributs anomenat "group_golf". Aquest grup contindrà el camp de golf de les activitats de tipus "Golf". Per controlar que aquest grup es mostri o no a l'usuari ens ajudarem del mòdul Conditional Fields[12]. Amb aquest mòdul podem controlar la visualització o no d'atributs o grups d'atributs depenent del valor d'un altre atribut (que sigui de tipus llistat d'opcions). D'aquesta manera assignarem que només es mostri aquest grup quan l'activitat sigui de tipus "Golf".

[12] Més informació sobre Conditional Fields a: http://drupal.org/project/conditional_fields



Il·lustració 12 - Configuració del grup "Golf" utilitzant Conditional Fields

- **Camp de golf:** En aquest punt del projecte encara no disposem del tipus de contingut "Camp de golf". Per poder assignar el camp a l'activitat primer crearem un tipus de contingut per al camp de golf. En aquesta mateixa iteració ja afegirem els atributs que necessitem. De moment, només necessitem que estigui creat el tipus de contingut.

  Recordem que un tipus de contingut al cap i a la fi està format per atributs específics afegits als genèrics de qualsevol node. Llavors quan es crea una activitat de tipus golf i afegim el camp de golf el que s'està realitzant és una relació entre 2 nodes diferents. Per poder crear relacions entre els nodes farem servir el mòdul Node Relationships[13] que amplia les funcionalitats del mòdul Node Reference (inclòs al CCK).

  Un cop explicat què necessitem per relacionar nodes afegirem al tipus de contingut activitat l'atribut "camp de golf" que serà de tipus "node reference". Llavors, un cop fet això, la relació estarà creada, com es pot veure gràficament gràcies a les eines del mòdul Node Relationships:

[13] Més informació a: http://drupal.org/project/noderelationships

Il·lustració 13 - Diagrama que mostra la relació en el tipus de contingut "Activitat" i "Camp de golf"

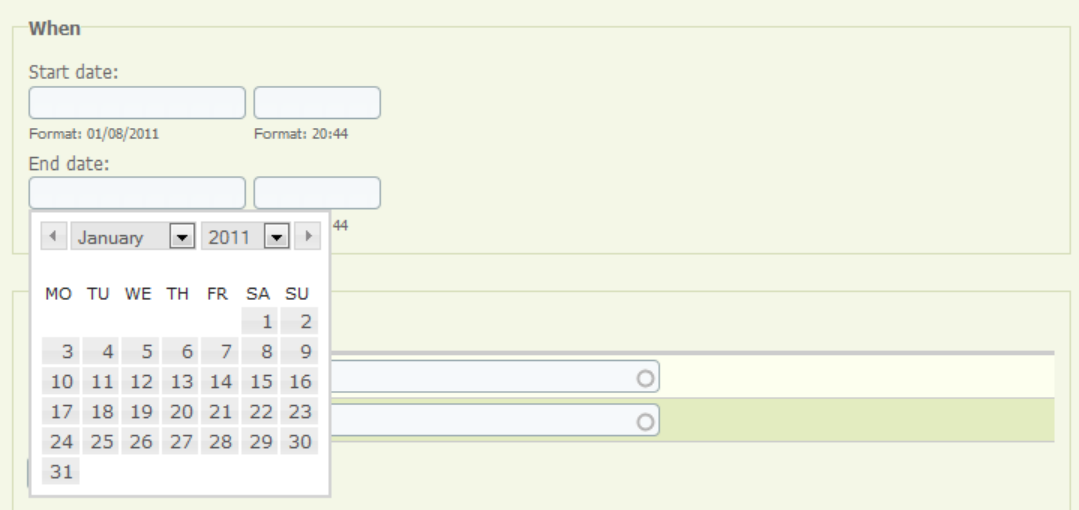
També gràcies al mòdul Node Relationships activarem l'opció de permetre crear i relacionar contingut. D'aquesta manera si al crear una activitat de tipus "Golf" i indicar el camp de golf ens adonem que no existeix, podem crear-lo des de la pantalla de creació de l'activitat alhora que queda marcat com el camp de golf de l'activitat. La forma d'introducció de les dades serà amb el widget autocomplete, de manera que el sistema mostrarà els camps que coincideixen amb el text que estem introduint.

Il·lustració 14 - Atribut "Camp de golf" de tipus Node Reference

- **Durada:** Es crearà el grup d'atributs "group_when" que inclou els atributs que fan relació a la ubicació temporal de l'activitat.
    - **Data d'inici:** Per crear camps de tipus data s'utilitzarà el mòdul Date[14]. Amb aquest mòdul podem crear atributs de tipus data definint la granularitat, que pot anar d'una data indicant només l'any fins a indicar la data en format dd/mm/YYYY h:m:s (és a dir, fins als segons). En el nostre cas, arribarem fins al

[14] Més informació del mòdul Date a: http://drupal.org/project/date

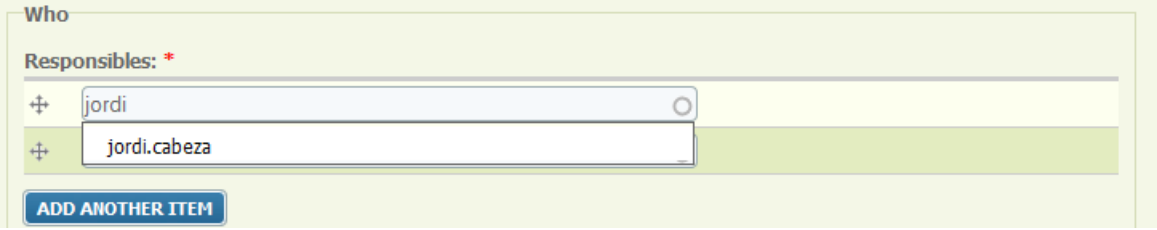
minut ja que una activitat no cal que comenci a una hora en punt. Escollirem que la data es pugui inserir mitjançant un calendari que apareixerà al clicar damunt el camp en el formulari, això es pot realitzar amb el mòdul Date Popup(inclòs a Date). El resultat és el següent:



Il·lustració 15 - Atribut de tipus data amb calendari pop up per introduir la data

- **Data fi:** Es crearà l'atribut d'igual forma que la data d'inici.

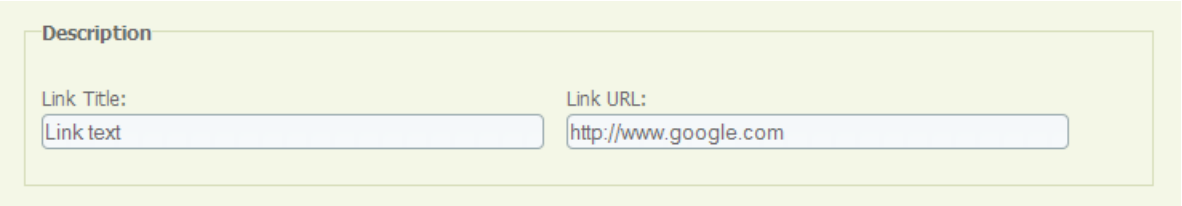
- **Qui:** Crearem el grup d'atributs "group_who" que inclou els atributs que fan relació als usuaris relacionats amb l'activitat.
  - **Responsables:** D'igual forma que poden crear relacions entre nodes, Drupal permet relacionar nodes amb usuaris mitjançant el mòdul User Reference (inclòs també al mòdul CCK). Per introduir la informació de la relació utilitzarem un altre cop el widget autocomplete. En aquest cas, indicarem que el nombre d'elements és indefinit, és a dir, pot haver-hi tants responsables de l'activitat com es desitgi. També indicarem que és obligatori, de manera que sempre s'ha d'indicar com a mínim un responsable.



Il·lustració 16 - Atribut de tipus User Reference de cardinalitat il·limitada

- **Participants:** Es crearà l'atribut d'igual forma que l'anterior amb l'excepció que no serà obligatori, ja que al crear l'activitat pot no haver-hi cap participant en aquell precís instant.

- **Descripció:** Crearem el grup d'atributs "group_description".
  - **Enllaç:** Per afegir aquest atribut utilitzarem el mòdul Link[15]. Aquest mòdul permet a l'usuari afegir atributs de tipus enllaç de forma molt fàcil, ja que automàticament crea al formulari d'edició un camp per afegir el text visible de l'enllaç i un altre per indicar la URL on ens portarà l'enllaç, tot controlant que sigui una URL amb format correcte. També s'encarrega de crear l'enllaç en HTML.

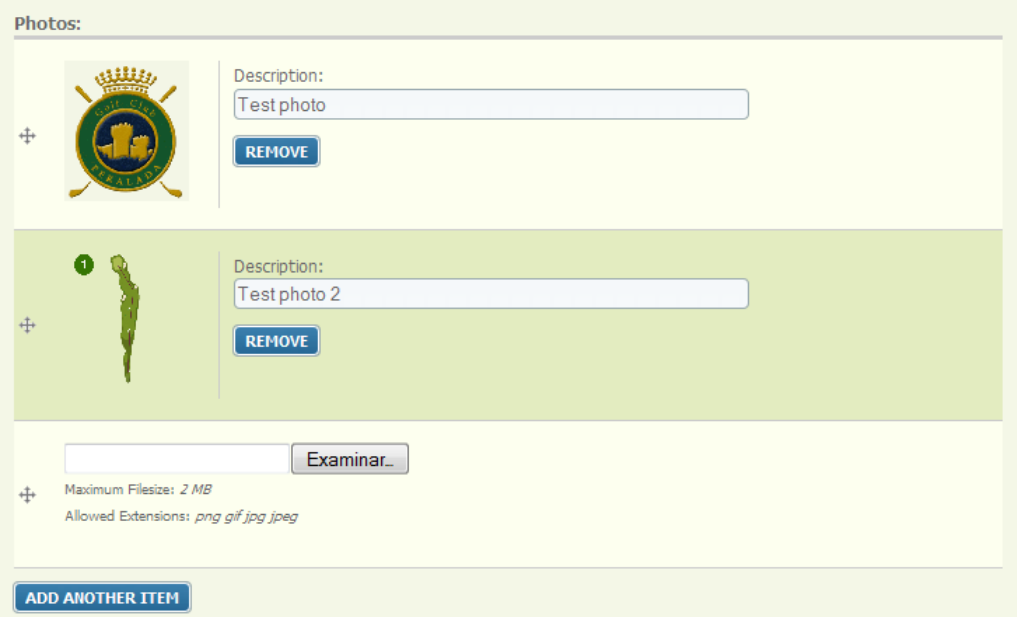


Il·lustració 17 - Atribut de tipus Link

- **Galeria de fotografies:** Per afegir una galeria de fotografies hem d'afegir 2 atributs al tipus de contingut. El primer serà l'atribut que permetrà pujar les fotos i el segon serà una galeria que mostrarà totes les fotos introduïdes en el format desitjat.
  Per a poder introduir fotografies crearem un camp de tipus "File". Aquests camps es poden crear gràcies al mòdul Filefield[16], que permet crear atributs de tipus fitxer, com poden ser documents o fotografies. Al afegir l'atribut podem definir les seves

[15] Més informació del mòdul Link a: http://drupal.org/project/link
[16] Més informació del mòdul Filefield a: http://drupal.org/project/filefield

propietats, com pot ser el número de fitxers que permeten adjuntar, les extensions de fitxers que acceptem,i en el cas d'imatges, les resolucions mínimes i màximes. En el nostre cas, permetre pujar qualsevol nombre d'imatges que siguin en format "png gif jpg jpeg" sense restriccions en la resolució. El resultat que ens trobem en el formulari d'edició és el següent:



Il·lustració 18 - Atribut de tipus Filefield

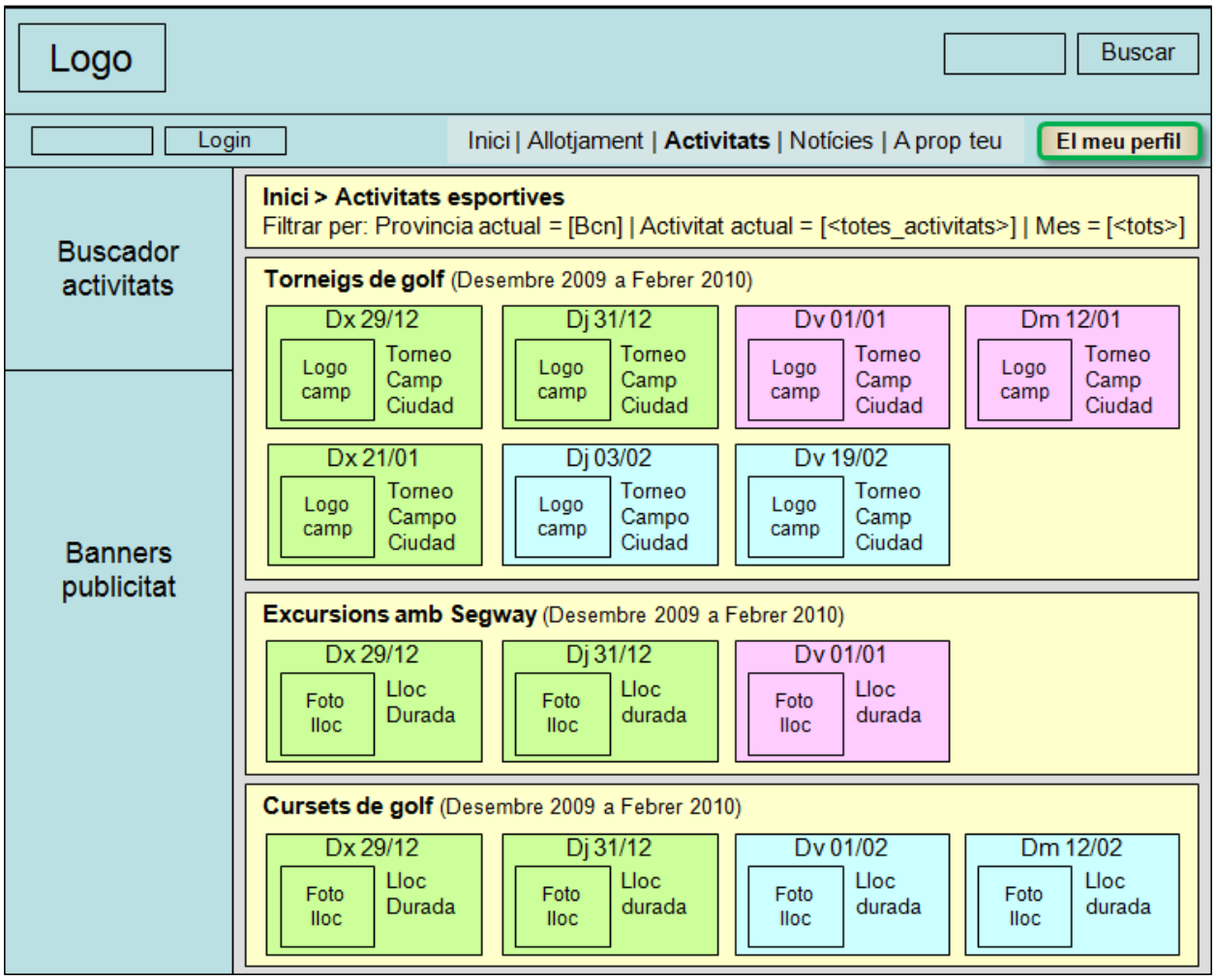
Per afegir la galeria utilitzarem el mòdul Viewfield[17], que permet crear atributs per als tipus de contingut que conté el resultat d'una vista. Tot i que encara no hem parlat de vistes a Drupal (ho farem en la següent funcionalitat), ens quedarem amb la idea bàsica que una vista és un llistat de contingut amb el format que s'indiqui. En el nostre cas serà el llistat de fotografies on es mostrarà la imatge en miniatura (thumbnail) que obrirà un popup amb l'ampliació. Aquest llistat l'obtindren de la vista Gallery, una vista ja inclosa al instal·lar el mòdul Viewfield.

[17] Més informació del mòdul Viewfield a : http://drupal.org/project/viewfield

4.2.2 Vista d'activitats agrupades per tipus

Amb aquesta funcionalitat es pretén donar a l'usuari l'oportunitat de visualitzar les activitats que tindran lloc ordenades per data i agrupades per tipus d'activitat. També es vol donar l'oportunitat que en cas d'haver-hi moltes activitats, l'usuari pugui filtrar les activitats per visualitzar aquelles en les quals es troba més interessat.

4.2.2.1Pantalla



Il·lustració 19 - Disseny preliminar de la funcionalitat

4.2.2.2Especificació detallada

Del disseny inclòs a l'apartat anterior es detallaran alguns aspectes:

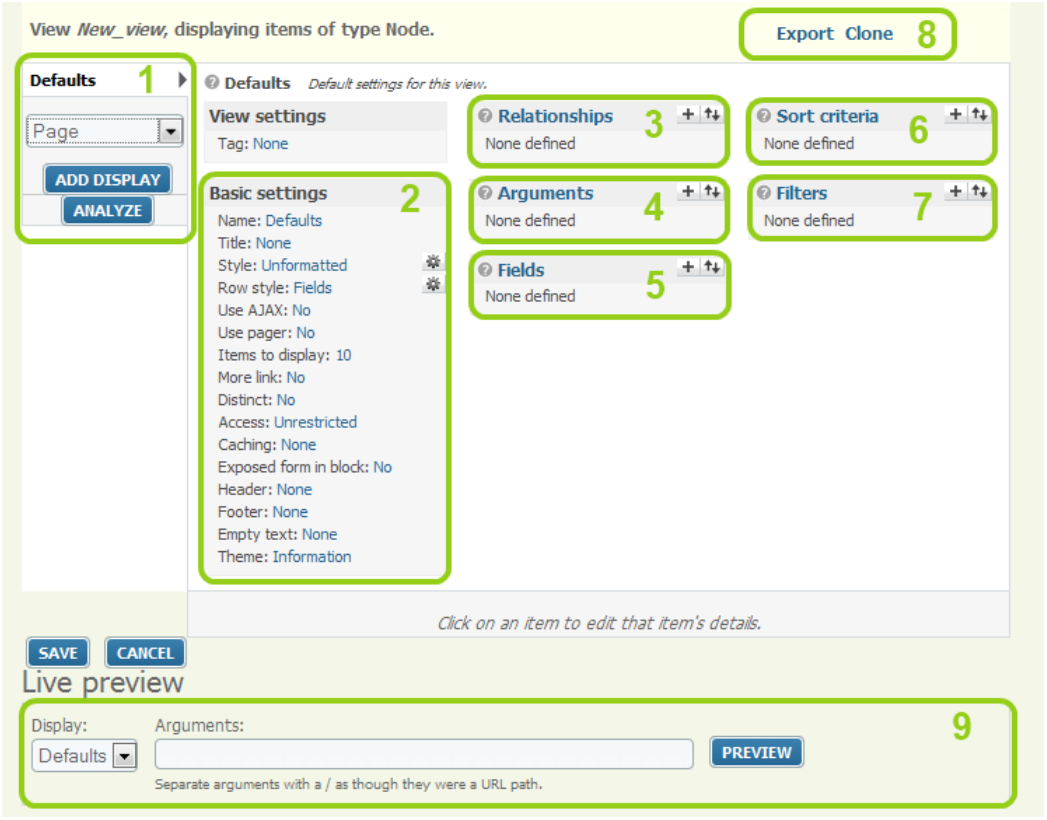
1- En la part superior de la pantalla s'ha de poder filtrar les activitats per tipus, província i data.

2- Les activitats han d'estar clarament agrupades pel tipus d'activitat.

3- Per cada activitat s'ha d'indicar el títol, la data, la població on és realitza, el camp de golf en cas de ser de tipus golf i una miniatura d'una de les fotos de cada activitat.

4.2.2.3Disseny de la funcionalitat

Per realitzar llistats de continguts en Drupal s'utilitza el mòdul Views[18]. Aquest mòdul permet als administradors definir llistats de continguts, anomenats vistes, controlant el contingut i la forma de presentar-ho. En essència, proporciona una interfície per accedir al contingut de la base de dades sense haver d'utilitzar el llenguatge SQL i alhora permet escollir com presentar la informació. A més a més, existeixen diferents mòduls que amplien les opcions de visualització i d'escollir la informació. Durant l'elaboració d'aquest projecte s'aniran presentant aquestes extensions conforme es vagin necessitant. A continuació es mostra l'entorn gràfic que proporciona Drupal per crear vistes definint els blocs dels quals consta:



Il·lustració 20 – Entorn de creació de vistes

[18] Més informació a: http://drupal.org/project/views

Seguint els números de la il·lustració 18:

1. En aquest bloc veiem les visualitzacions d'una vista. Una visualització d'una vista és la ubicació que li donem. Pot ser una pàgina (node de tipus pàgina), un bloc que podrem afegir en qualsevol lloc del portal mitjançant l'administració de blocs del portal (en Drupal el blocs són seccions de contingut que podem afegir a les diferents parts en que es divideix la pantalla com pot ser el menú, el cercador de cases rurals, etc..). També permet altres opcions que de moment no utilitzarem (com visualitzar la vista en forma de calendari o de web feed per rebre informació).
2. El bloc de "Basic settings" permet modificar la següent informació general de la vista. Les opcions més importants són les següents:
    a. **Nom de la vista** (serveix per identificar)
    b. **Títol de la vista** (es veurà per pantalla)
    c. **Estil de la vista**: Podem escollir si volem que el contingut es presenti en format de taula, de llista html, de graella, etc. Mitjançant mòduls podem afegir més opcions de visualització.
    d. **Estil de cada fila**: Permet escollir si volem mostrar els camps escollits per cada element o tota la informació del node.
    e. **Usar AJAX**: Permet escollir si volem utilitzar AJAX per fer que quan utilitzem el paginador o els possible filtres de cerca no s'hagi de recarregar la pàgina.
    f. **Usar paginador**: Permet escollir si s'utilitza o no paginador per mostrar el contingut.
    g. **Elements a visualitzar**: Permet escollir els elements que es mostren per cada pàgina. Podem escollir que es mostrin tots els elements.
3. El bloc "Relationships" permet definir relacions de contingut d'igual manera que podem fer "joins" en SQL. Per exemple, si estem llistant activitats i aquestes estan relacionades amb els camp de golf, haurem de crear una relació si volem accedir a la informació del camp.
4. El bloc "Arguments" permet definir valors que alteren el resultat d'una consulta però que no són fixos. Per exemple, si volem llistar les activitats que ha creat un usuari

podem indicar en “Arguments” que utilitzi el valor del usuari de la sessió. Equivalen a condicions dins el “where” d’una sentencia SQL.

5. El bloc “Fields” permet escollir els atributs que volem mostrar per cada element del llistat d’igual forma que fem en el “select” d’una sentencia SQL.
6. El bloc “Sort criteria” permet definir criteris d’ordenació del contingut que és llistarà. És l’homòleg del “order by” en una sentencia SQL.
7. Com a últim bloc per definir el contingut del llistat es troba el bloc “Filters”, que permet afegir condicions com és fa en el “where” d’una sentencia SQL. Per exemple, si volem llistar activitats haurem d’afegir el filtre que indiqui que només volem nodes que siguin del tipus de contingut “Activitat”.
8. A la part superior, trobem les opcions de clonar i exportar, que ens poden ser útils si tenim una vista ja creada i hem de crear una de molt similar o si l’hem d’afegir en una altra instal·lació de Drupal respectivament.
9. Finalment, Drupal ens permet fer proves de la vista que anem construint clicant al botó “Preview”. D’aquesta manera ens anem assegurant a cada pas que la vista que s’està construint satisfà les necessitats.

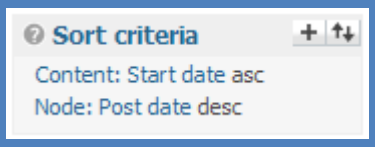
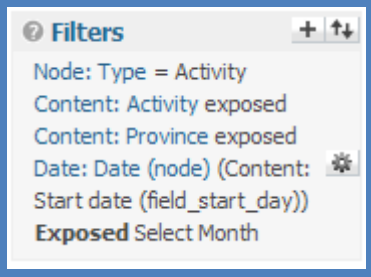
Un cop introduït l’entorn de treball per crear la vista ja es pot presentar la feina realitzada. Per fer-ho s’utilitzarà el mateix procediment que fins ara. S’indicarà una il·lustració amb la configuració realitzada i a continuació es detallarà la configuració de cada bloc.

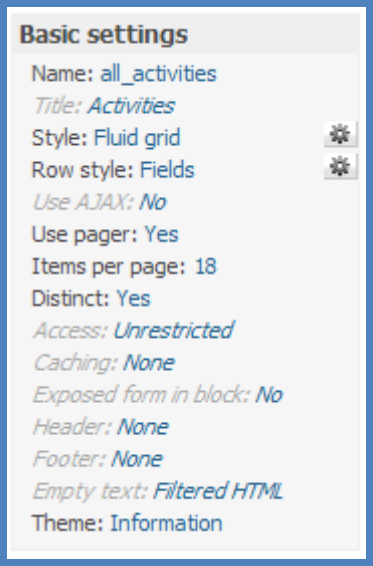
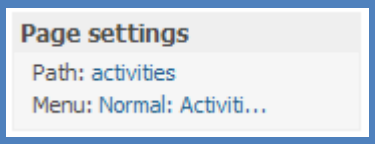
Il·lustració 21 – Configuració de la vista d'activitats agrupades per tipus

Per realitzar la vista d'activitats s'ha decidit crear una vista de nodes (també es poden crear vistes d'usuaris) afegint una visualització de tipus pàgina. D'aquesta manera és pot assignar un path per poder accedir a la vista així com afegir-la com a opció del menú de navegació.

En la següent taula veurem en detall la configuració de cada bloc de la vista. Ens centrarem primer en els blocs que permeten obtenir el contingut.

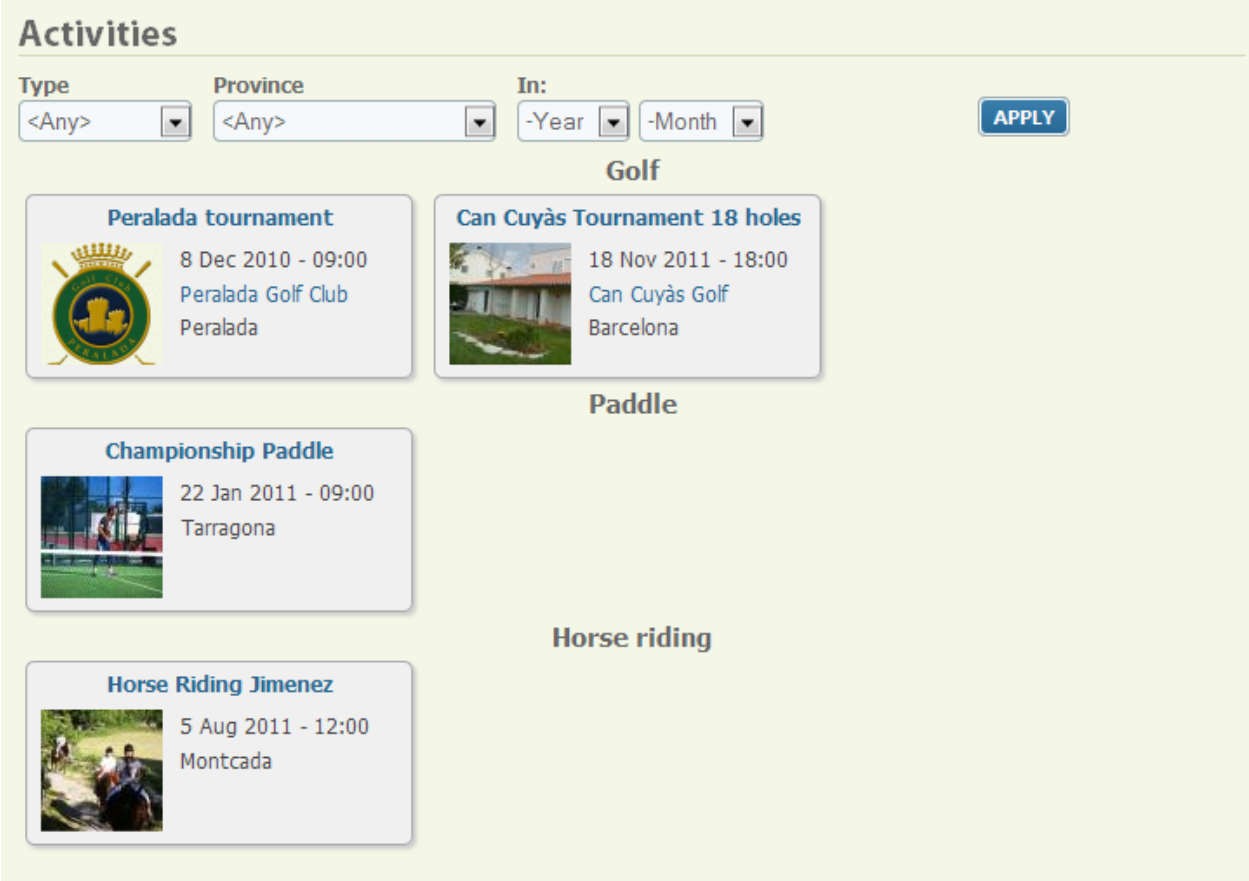
Bloc de configuració	Configuració
Fields Node: Title Content: Photo miniThumb image linked to node Content: Start date Short Content: Campo de Title (link) Content: Activity As Text Content: City Default	Els camps que s'han afegir a la vista són: - Títol del node (de l'activitat) - Primera fotografia afegida a l'activitat en format miniatura. S'ha escollit que al clicar damunt la imatge ens porti a veure la informació de l'activitat

	- Data d'inici de l'activitat en format abreviat (ex: 8 Dec 2010 - 09:00) - Nom del camp de golf que fa d'enllaç per veure la informació del camp (en cas de no ser de tipus golf no es veurà aquest camp) - Tipus de l'activitat (no es mostrarà directament ja que l'hem exclòs) però ens servirà per poder agrupar el contingut del mateix tipus.
	El resultat de la vista estarà ordenat per la data d'inici en primer terme de forma ascendent (de més antiga a més nova) i en cas d'empat, per la data de creació del node de forma descendent (primer el més antic). No serà habitual que dos activitats del mateix tipus comencin a la mateixa hora del mateix dia, de manera que amb el primer criteri d'ordenació hauríem de tenir suficient.
	S'ha definit un filtre per fer definir que els nodes han de ser de tipus "Activitat". Per altra banda, s'han creat filtres on l'usuari serà qui indicarà el seu valor per poder filtrar els resultats per defecte. Per fer-ho s'han de marcar com "exposed", així quedaran exposats en un formulari. Aquest filtres són: - Permetre escollir el tipus d'activitat a mostrar - Permetre escollir la província de les activitats - Permetre escollir el més de les activitats

	En el bloc de "Basic settings" hem indicat que l'estil de la vista sigui "Fluid grid". Per poder indicar aquest estil s'ha d'afegir al sistema el mòdul Views Fluid Grid[19], que permet ficar cada contingut del llistat en una casella dins d'una graella per obtenir un aspecte com el que es detalla al disseny preliminar en la il·lustració 17. Un cop escollit hem configurat les dimensions de la capsa de cada contingut i hem indicat que el camp que indica el tipus d'activitat serveixi per agrupar el contingut (només es pot agrupar per camps afegit en el bloc "Fields"). Altres elements que hem configurat són permetre la paginació de resultats amb 18 elements per pàgina. Hem escollit un múltiple de 3 ja que en cada fila de la graella hi caben 3 elements, de manera que es mostraran 6 files per pàgina.
	Com s'ha escollit que la visualització de la vista sigui en una pàgina, el mòdul Views ens permet configurar el path de la pàgina així la pàgina en el menú de navegació. El path escollit ha sigut "activities", de manera que si en el navegador posem la url del nostre site url_site_drupal/activities podrem accedir a la vista. Altrament, al incloure la opció en el menú, l'usuari podrà accedir fàcilment a la vista.

Un cop configurada la vista, i asegurat que els filtres funcionen correctament, ja es pot accedir a la pàgina a través del menú (opció activities).

[19] Més informació del mòdul a: http://drupal.org/project/views_fluidgrid



Il·lustració 22 – Vista de les activitats agrupades per tipus

4.2.3 Gestió de camps de golf

En la funcionalitat del punt 4.2.1 Gestió d'activitats hem creat el tipus de contingut "Camp de golf" ja que el necessitàvem per poder crear l'atribut de l'activitat que fa referència al camp de golf. En la funcionalitat anterior únicament s'havia creat el tipus de contingut sense cap atribut específic, llavors només conté els atributs bàsics de tot node: identificador numèric, títol i camp de text per introduir la descripció. Amb aquests camps únicament podem definir el títol i la descripció del camp de golf. Per tant, ara cal definir la resta d'atributs.

4.2.3.1Especificació detallada

El client no ens va indicar en primera instancia cap preferència en relació a la informació que ha de contenir el camp de golf. El que sí va indicar va ser com volia veure el resultat d'una partida de golf, que havia de ser en format de targeta de golf, on s'inclou tota la informació relativa als

18 forats d'un camp de golf. També es va decidir poder incloure fotografies del camp, així com una imatge del logotip del camp.

Per tant, finalment el tipus de contingut "Camp de golf" contindrà la següent informació:

Camp de golf:

- **Títol:** Nom del camp de golf.
- **Descripció:** Text descriptiu del camp de golf.
- **Logotip/Imatge principal:** Imatge amb el logotip del camp
- **Galeria d'imatges:** Galeria d'imatges relatives al camp
- **Per cadascun dels 18 forats:**
    - **Número del forat:** S'indicarà el número (entre l'1 i el 18) del forat a dins del camp.
    - **Par:** Número de cops ideals amb els que s'ha de ficar la pilota de golf dins el forat. El funcionament de la puntuació[20] en el golf ve donat pel numero de cops amb que s'aconsegueix ficar la pilota dins del forat i depèn del par del camp.
    - **Longitud:** Distància en metres des de el punt de sortida fins al forat.
    - **Handicap:** El handicap d'un forat serveix per identificar la dificultat del forat. És un número entre l'1 i el 18. El forat amb handicap 18 serà el més fàcil de realitzar el seu recorregut amb el par indicat, en canvi, el forat amb handicap 1 serà el més difícil. Podem dir que ordena els forats per dificultat de forma decreixent.

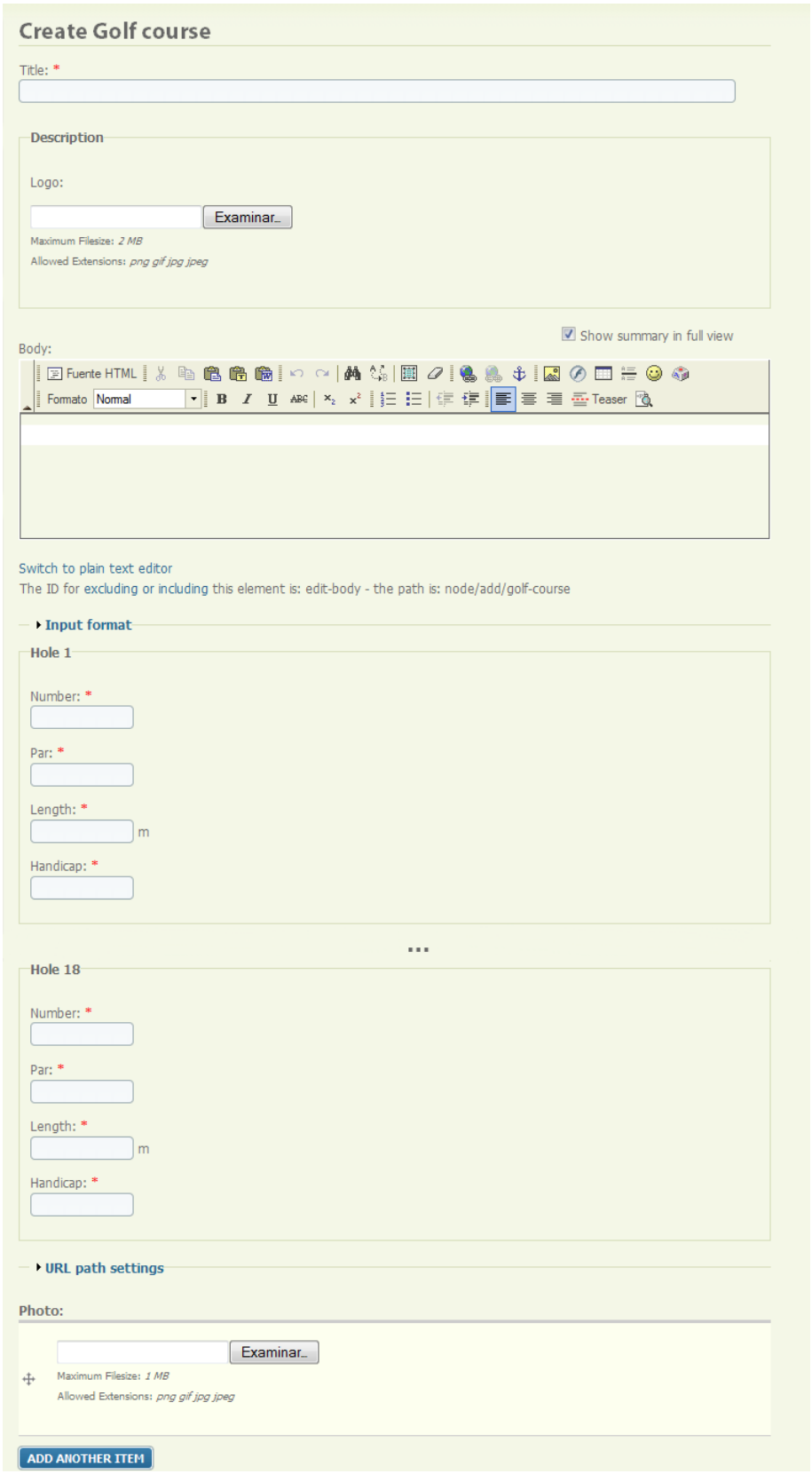
4.2.3.2Disseny de la funcionalitat

En aquest cas no caldrà definir el funcionament del mòdul CCK vist anteriorment i aquesta secció quedarà limitida a definir el procés de creació dels atributs del camp de golf. Tenint en compte que el títol i la descripció ja han estat definits, en centrarem en explicar com s'ha creat la resta d'atributs.

[20] Més informació sobre la puntuació en el golf: http://es.wikipedia.org/wiki/Golf#Puntuaci.C3.B3n

- **Logotip:** Per a poder introduir el logotip crearem un camp de tipus "File" de cardinalitat 1 (ja que només volem afegir un logotip) sense restricció en la resolució de la imatge i permetent els formats "png gif jpg jpeg".
- **Galeria d'imatges:** Afegirem una galeria d'imatges de la mateixa forma que ha sigut afegida pel tipus de contingut "Activitat" creant primer el camp de tipus "File" on poder pujar les imatges i després el camp de tipus Viewfield per incloure la vista amb la galeria d'imatges.
- **Forat 1. . .18:** Per cada forat de l'1 al 18 s'ha creat un grup que conté els atributs de cada forat. S'ha decidit adoptar aquesta opció després de descartar altres com crear un tipus de contingut per al forat i relacionar-los amb el camp o utilitzar alguna taula/matriu per agrupar tota la informació del camp en un únic forat. L'opció desenvolupada permet un millor accés i distribució de les dades que ens serà útil al desenvolupar funcionalitats de les següents iteracions.
    - **Número del forat X:** S'afegeix un camp numèric (Integer) de tipus obligatori per indicar el número del forat.
    - **Par forat X:** S'afegeix un camp numèric (Integer) de tipus obligatori per indicar el par del forat.
    - **Longitud del forat X:** S'afegeix un camp numèric (Integer) de tipus obligatori per indicar el longitud en metres del forat. Indicarem que afegeixi com a sufix la lletra "m" per que a l'hora de visualitzar es pugui veure que la longitud ve donada en metres.
    - **Handicap del forat X:** S'afegeix un camp numèric (Integer) de tipus obligatori per indicar el handicap del forat.

Després d'afegir els atributs es pot veure com queda el formulari de creació/edició de camps de golf en la següent il·lustració. Per a futurs desenvolupaments, s'hauria d'estudiar la possibilitat de modificar el formulari evitant que un usuari hagi d'omplir un formulari tan llarg, tot compactant els camps del formulari.



Il·lustració 23 – Vista del formulari de creació/edició d'un camp de golf

5 Iteració 2

5.1 Llista d'activitats de la iteració

5.1.1 Gestió de partides de golf

1- Estudiar la informació que necessitem guardar d'una partida de golf.
2- Crear un nou tipus de contingut per guardar informació sobre les partides de golf tenint en compte que en següents iteracions s'haurà d'enregistrar estadístiques.
3- Permetre relacionar les partides de golf a un camp de golf que s'hagi inserit prèviament.

5.1.2 Vista de les activitats on participo

1- Crear una vista que permeti veure les activitats on un usuari participa o ha participat.
2- Mostrar el resultat en forma de graella que faciliti la lectura i comprensió de la informació

5.1.3 Vista de les activitats que gestiono

1- Crear una vista que permeti veure les activitats que organitza un usuari.
2- Mostrar el resultat en forma de taula que faciliti la lectura i comprensió de la informació.

5.1.4 Creació de plantilles per visualitzar activitats, partides i camps de golf

1- Estudiar com poder modificar les plantilles que Drupal ofereix a l'hora de mostrar tipus de continguts sense haver de tocar el tema que utilitza el portal.
2- Crear una plantilla per mostrar el contingut de tipus "Activitat".
3- Crear una plantilla per mostrar el contingut de tipus "Partida de golf".
4- Crear una plantilla per mostrar el contingut de tipus "Camp de golf".

5.2 Anàlisi i disseny de les funcionalitats

5.2.1 Gestió de partides de golf

En funcionalitats de la iteració anterior hem creat els tipus de contingut "Activitat" i "Camp de golf". En aquest punt crearem el tipus de contingut "Partida de golf" per permetre als usuaris poder enregistrar el resultat de les seves partides.

5.2.1.1Especificació detallada

El client ens va indicar que volia visualitzar el resultat d'una partida de golf de forma semblant al que es pot veure en una targeta de golf. En la següent il·lustració es pot veure el contingut d'una targeta de golf.

HOYO	1	2	3	4	5	6	7	8	9	IN	10	11	12	13	14	15	16	17	18	OUT	TOTAL
DISTANCIA	70	60	60	60	40	60	57	35	35	477	70	60	60	50	60	57	60	60	35	512	989
PAR	**3**	**3**	**3**	**4**	**3**	**3**	**3**	**3**	**3**	**27**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**27**	**54**
HANDICAP	9	3	5	11	1	7	13	15	17		10	4	6	12	2	8	14	16	18		
	3	2	1	1	4	5	6	7	8	37											

Il·lustració 24 – Exemple de targeta de golf. Font:

En la il·lustració es pot veure que una targeta consta de tota la informació del camp de golf (que ja tenim enregistrada mitjançant el tipus de contingut "Camp de golf") més la puntuació dels jugadors. El client ens ha informat que les partides han de tenir entre 1-4 jugadors.

Amb aquesta informació ja podem definir el tipus de contingut "Partida de golf".

Partida de golf:

- **Títol:** Nom de la partida.
- **Camp de golf:** Indicarà el camp on es realitza la partida
- **Jugadors:**
    - **Jugador1:** Usuari que jugarà la partida ocupant la primera línia de la targeta
    - **Jugador2:** Usuari que jugarà la partida ocupant la segona línia de la targeta
    - **Jugador3:** Usuari que jugarà la partida ocupant la tercera línia de la targeta

    - **Jugador4:** Usuari que jugarà la partida ocupant la quarta línia de la targeta
- **Puntuació jugador1:**
    - **Puntuació forat 1:** Puntuació del jugador2 al forat 1
    - ... (fins al forat 18)
    - **Puntuació forat 18:** Puntuació del jugador2 al forat 18
- **Puntuació jugador2:**
    - **Puntuació forat 1:** Puntuació del jugador2 al forat 1
    - ... (fins al forat 18)
    - **Puntuació forat 18:** Puntuació del jugador2 al forat 18
- **Puntuació jugador3:**
    - **Puntuació forat 1:** Puntuació del jugador3 al forat 1
    - ... (fins al forat 18)
    - **Puntuació forat 18:** Puntuació del jugador3 al forat 18
- **Puntuació jugador4:**
    - **Puntuació forat 1:** Puntuació del jugador4 al forat 1
    - ... (fins al forat 18)
    - **Puntuació forat 18:** Puntuació del jugador4 al forat 18

5.2.1.2Disseny de la funcionalitat

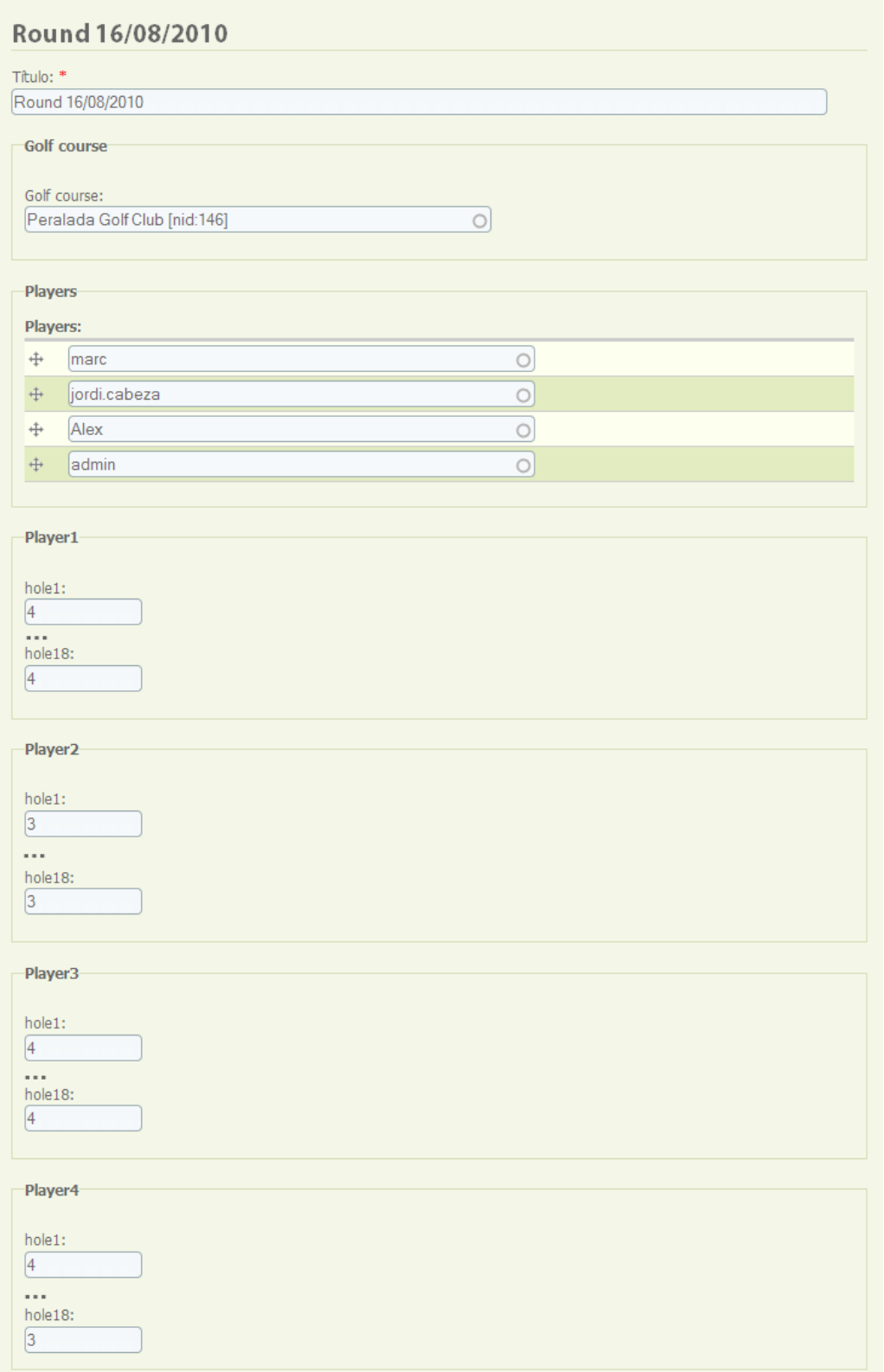
D'igual forma que en el camp de golf, s'ha decidit utilitzar l'estructura d'un atribut per cada casella de la puntuació de la targeta de golf ja que facilitarà la feina a l'hora de crear la plantilla per visualitzar el contingut així com a l'hora de calcular les estadístiques.

A continuació mostrem la feina realitzada en la creació dels atributs de la partida de golf, partint de que el títol ja ens ve donat per l'estructura de tot node.

- **Camp de golf:** S'afegirà un atribut de tipus "Node reference" per relacionar la partida de golf amb el camp de golf. La cardinalitat de l'atribut serà d'un únic element ja que una partida te lloc només a un camp. S'utilitzarà el widget autocomplete que mostrarà els camps que coincideixin amb el text introduït

- **Jugadors:**
    - **Jugador1...4:** Es crearà una camp de tipus "User reference" per permetre vincular els usuaris com a jugadors de la partida. També s'utilitzarà el widget autocomplete per introduir els usuaris i en aquest cas la cardinalitat estarà compresa entre 1..4 per permetre un màxim de 4 jugadors.
- **Puntuació jugador1:** Es crearà un grup per unificar les puntuacions del jugador1 en un mateix bloc
    - **Puntuació forat 1..18:** S'afegiran 18 atributs de tipus "Integer" per indicar la puntuació del jugador en cada forat del camp .
- **Puntuació jugador2:** Es crearà un grup per unificar les puntuacions del jugador2 en un mateix bloc
    - **Puntuació forat 1..18:** S'afegiran 18 atributs de tipus "Integer" per indicar la puntuació del jugador 2 en cada forat del camp .
- **Puntuació jugador3:** Es crearà un grup per unificar les puntuacions del jugador3 en un mateix bloc
    - **Puntuació forat 1..18:** S'afegiran 18 atributs de tipus "Integer" per indicar la puntuació del jugador 3 en cada forat del camp .
- **Puntuació jugador4:** Es crearà un grup per unificar les puntuacions del jugador4 en un mateix bloc
    - **Puntuació forat 1..18:** S'afegiran 18 atributs de tipus "Integer" per indicar la puntuació del jugador 4 en cada forat del camp .
    - del jugador4 al forat 18
    - **Handicap del forat X:** S'afegeix un camp numèric (Integer) de tipus obligatori per indicar el handicap del forat.

La següent il·lustració ens mostra com queda el formulari de creació/edició de partides de golf. D'igual forma que per al camp de golf, s'hauria d'estudiar la possibilitat de modificar el formulari evitant que un usuari hagi d'omplir un formulari tan llarg i pessat, tot compactant els camps del formulari facilitant l'entrada de dades. Els punts suspensius substituixen els atributs de la resta de la puntuació de cada jugador per a què la imatge no sigui excessivament llarga.

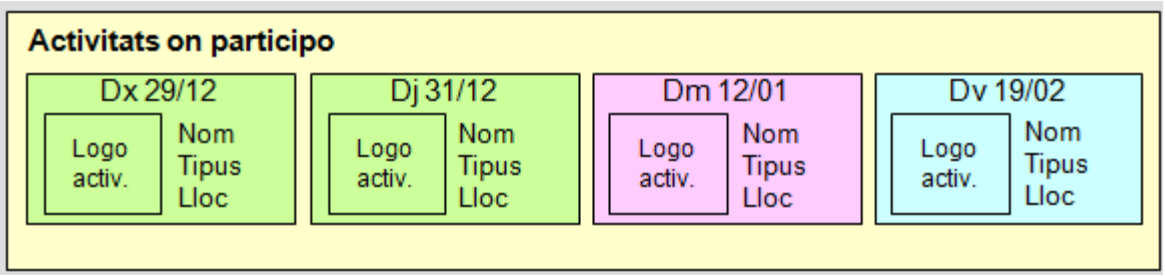


Il·lustració 25 – Vista del formulari de creació/edició d'una partida de golf

5.2.2 Vista de les activitats on participo

Amb aquesta funcionalitat es pretén donar a l'usuari de forma gràfica i visual la informació de les activitats on ell ha participat o te previst participar.

5.2.2.1Pantalla



Il·lustració 26 - Disseny preliminar de la vista "Activitats on participo"

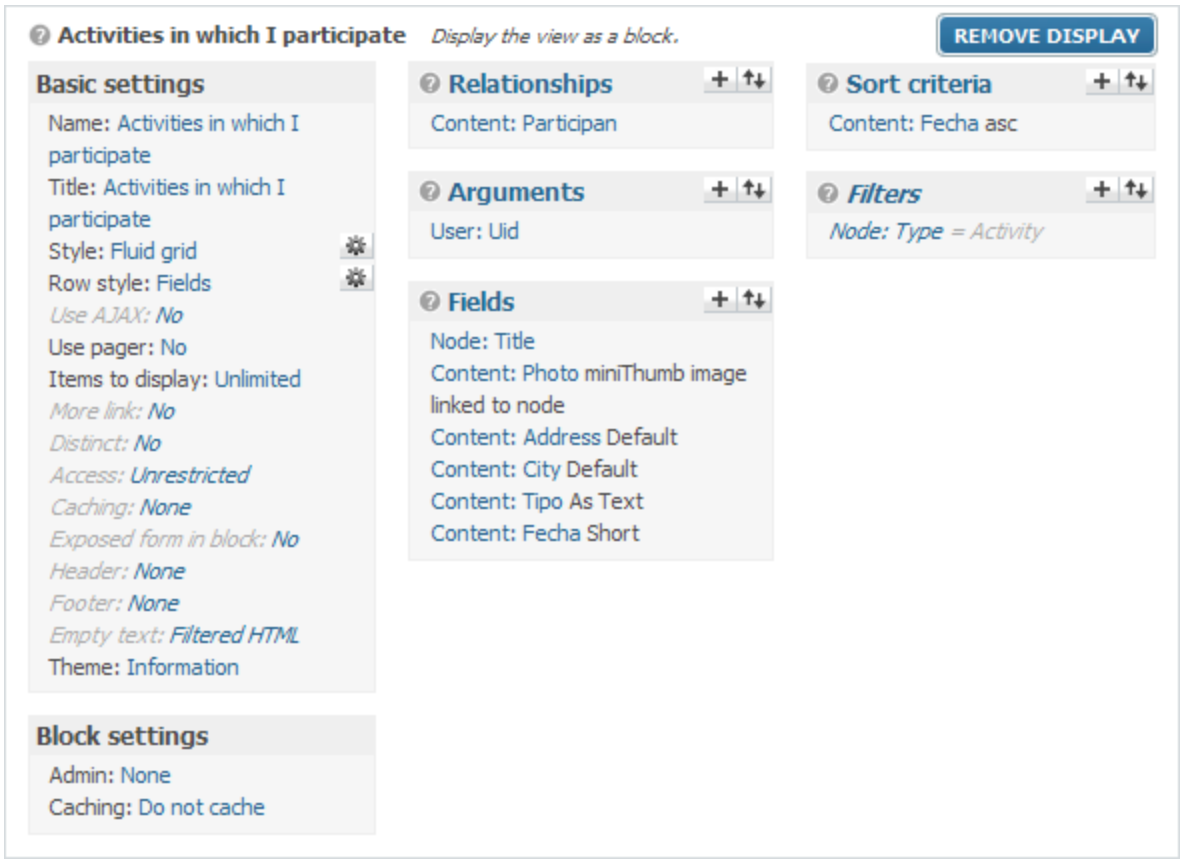
5.2.2.2Especificació detallada

Del disseny inclòs a l'apartat anterior obtenim:

1- Les activitats han d'estar ordenades per data de forma ascendent. De més antiga a més nova.
2- Per cada activitat s'ha d'indicar el títol, la data, el tipus d'activitat, la població on és realitza i una miniatura d'una de les fotos de cada activitat.

5.2.2.3Disseny de la funcionalitat

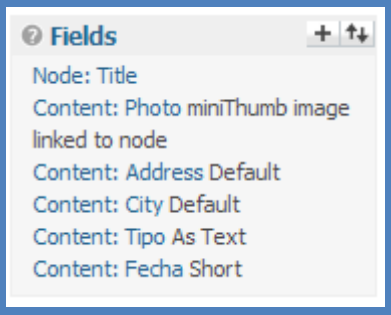
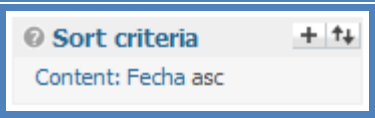
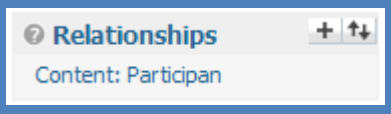
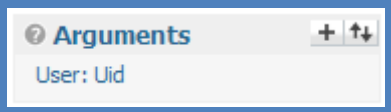
Per detallar la realització de la funcionalitat s'indicarà una il·lustració amb la configuració realitzada i a continuació es detallarà la configuració de cada bloc.

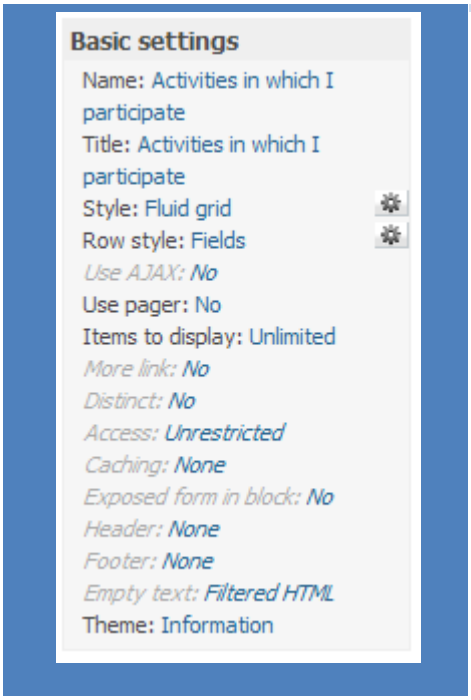


Il·lustració 27 – Configuració vista d'activitats on participo

En aquest cas, a diferencia de la vista d'activitats creada en la iteració anterior, aquesta serà de tipus "block". D'aquesta manera no tindrà una ubicació definida en el portal, sinó que la podrem incloure com un bloc més en qualsevol lloc del portal.

En la següent taula es veu en detall la configuració de cada bloc de la vista. Ens centrarem primer en els blocs que permeten obtenir el contingut.

Bloc de configuració	Configuració
	Els camps que s'han afegir a la vista són: - Títol del node (de l'activitat) - Primera fotografia afegida al camp activitat en format miniatura. Al clicar damunt la imatge ens porta a veure el node - Tipus de l'activitat - Data d'inici de l'activitat en format abreviat (ex: 8 Dec 2010 - 09:00)
	El resultat de la vista estarà ordenat per la data d'inici de forma ascendent (de més antiga a més nova).
	S'ha definit un filtre per fer definir que els nodes han de ser únicament de tipus "Activitat".
	Per poder escollir només les activitats en que participem haurem de crear primer una relació en el bloc "Relationships" amb el camp "Participants" que hem definit al crear l'activitat i que és on es guarden els usuaris que participen en l'activitat. Escollirem l'opció de quedar-nos amb tots el participants.
	En el bloc "Relationships" hem obtingut tots els participants de l'activitat. Ara el que s'ha de realitzar és escollir només aquelles activitats on l'usuari que està veient el resultat de la vista forma part dels participants. Per fer-ho afegirem un argument en el bloc "Arguments" que sigui l'identificador de l'usuari de la sessió activa i indicarem que utilitzi la relació creada en el bloc anterior.



Finalment, en el bloc "Basic settings" s'ha tornat a indicar que l'estil de la vista sigui "Fluid grid" amb la diferència que en aquest cas no s'ha escollit cap camp per agrupar el contingut.

Altres elements que s'han configurat han sigut no permetre la paginació i mostrar tots els elements en la mateixa pàgina.

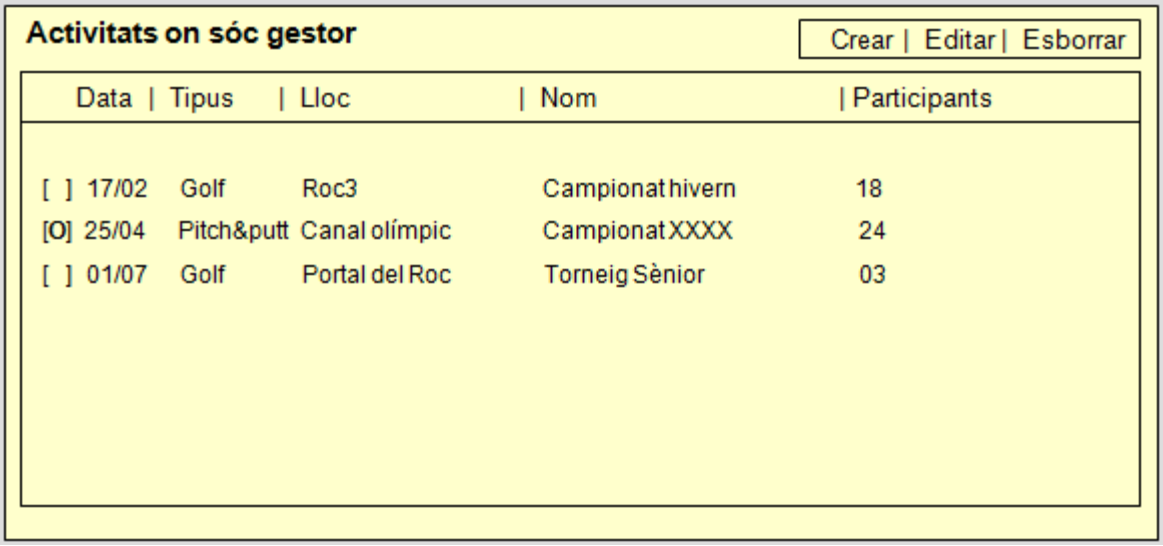
Un cop configurada la vista i fetes les proves corresponents el resultat que obtenim és el següent:

Il·lustració 28 – Vista de les activitats on participo

5.2.3 Vista de les activitats que gestiono

Amb aquesta funcionalitat es pretén que l'usuari pugui controlar les activitats que ha creat de forma fàcil i intuïtiva alhora que pot veure el número d'usuaris que s'han apuntat a l'activitat.

5.2.3.1Pantalla



Il·lustració 29 - Disseny preliminar de la vista "Activitats que gestiono"

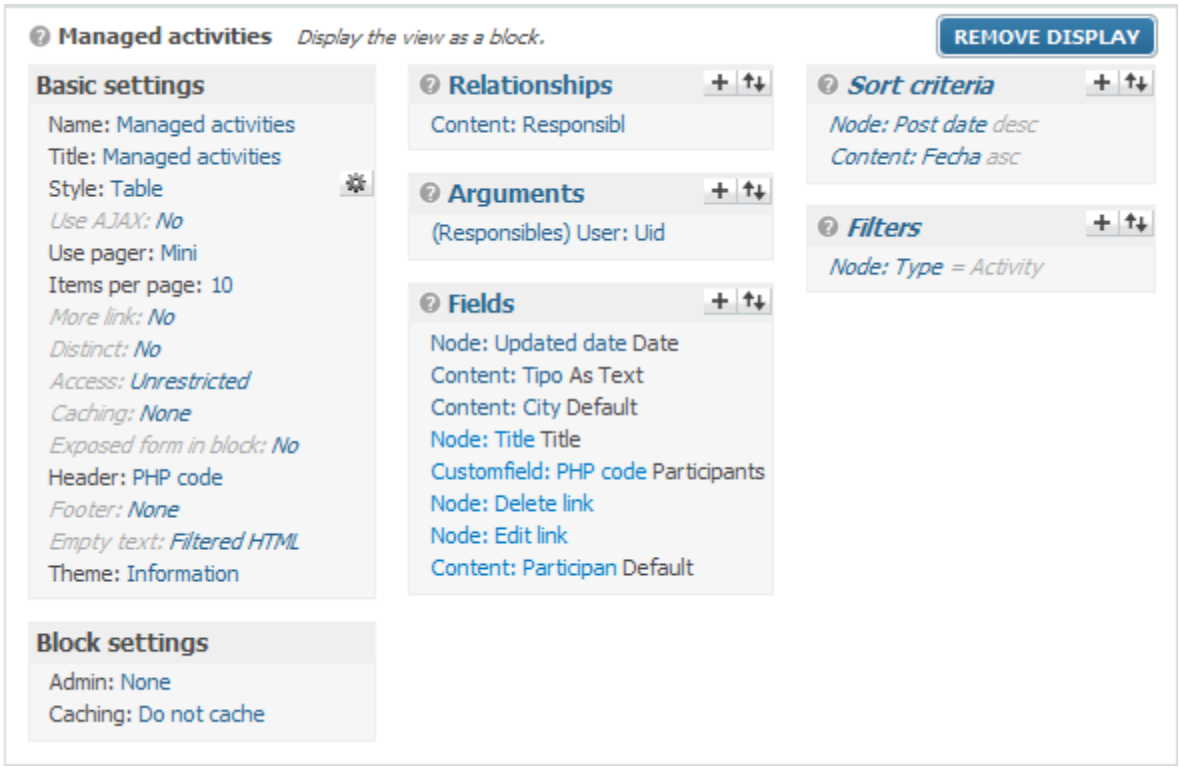
5.2.3.2Especificació detallada

Del disseny inclòs a l'apartat anterior obtenim:

1- El resultat de la vista s'ha de mostrar en format de taula.
2- Per cada activitat s'ha d'indicar la data de creació, el tipus d'activitat, el lloc on es realitza, el seu títol i el número de participants
3- S'ha de permetre un accés per poder crear noves activitats
4- S'ha de permetre accedir directament a l'opció d'editar o esborrar les activitats

5.2.3.3Disseny de la funcionalitat

Per detallar la realització de la funcionalitat s'indicarà una il·lustració amb la configuració realitzada i a continuació es detallarà la configuració de cada bloc.



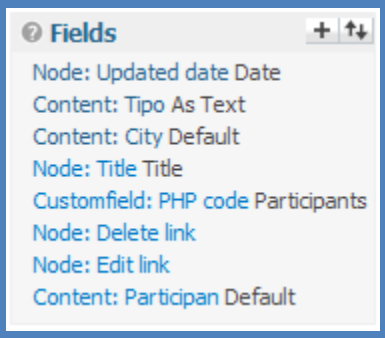
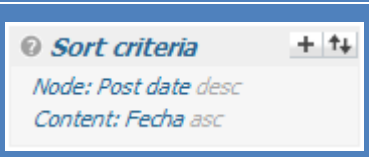
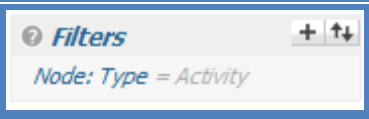
Il·lustració 30 – Configuració vista d'activitats que gestiono

D'igual forma que la vista creada a la funcionalitat anterior, aquesta també serà de tipus "block". Quan creem el perfil d'usuari en les següents iteracions ubicarem aquests dos blocs en la seva secció corresponent. Mentrestant romandran com a blocs sense ubicació definida en el sistema.

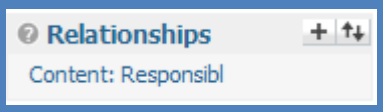
Per realitzar aquesta vista hem utilitzat un nou mòdul. El mòdul en qüestió és Views Custom Field[21]. Aquest mòdul ens permet definir camps personalitzat en el mòdul Views com pot ser el número de cada element de la llista (si volem una llista numerada) o introduir codi PHP en el camp. Per a la nostra vista aquest mòdul ens permetrà obtindre el número de participants de cada activitat mitjançant codi PHP.

A continuació mostrarem una taula amb els detalls de configuració de cada bloc de la vista.

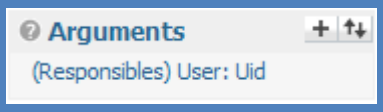
[21] Més informació del mòdul Views Custom Field a: http://drupal.org/project/views_customfield

<table>
<tr><th>Bloc de configuració</th><th>Configuració</th></tr>
<tr><td> 
</td><td>Els camps que s'han afegir a la vista són:
- Data de l'ultima edició del node (creació o edició) en format dd/mm/YYYY
- Tipus de l'activitat
- Població de l'activitat
- Títol del node (de l'activitat)
- Participants de la relació. Aquest camp indicarem que quedi ocult i no es mostri. L'afegim perquè a partir d'aquest camp podem obtenir el número de participants.
- Camp de tipus "customfield" (creat gràcies al mòdul Views Custom Field) que mitjançant PHP obtenim el número de participants:
<?php
print
count(
$data->node_data_field_participantes_field_participantes_uid
);
?>
Degut a que el camp ocult dels participants queda expressat en un Array, aplicant la funció count()[22] de PHP podem obtenir el número de participants.</td></tr>
<tr><td> 
</td><td>El resultat de la vista estarà ordenat per la data d'edició del node de forma descendent en primer terme seguit de la data de l'activitat, també en ordre descendent.</td></tr>
<tr><td> 
</td><td>S'ha definit un filtre per fer definir que els nodes han de ser únicament de tipus "Activitat".</td></tr>
</table>

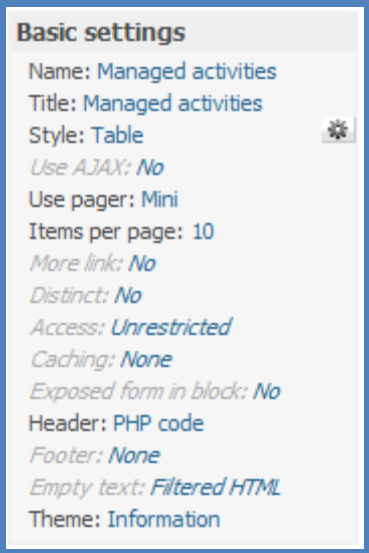
[22] Més informació de la funció count() a: http://php.net/manual/es/function.count.php



Per poder escollir només les activitats que gestionem haurem de crear primer una relació en el bloc "Relationships" amb el camp "Responsibles" que hem definit al crear l'activitat i que és on es guarden els usuaris que gestionen l'activitat. Escollirem l'opció de quedar-nos amb tots els usuaris de la relació.



En el bloc "Relationships" hem obtingut tots els participants de l'activitat. Ara el que s'ha de realitzar es escollir només aquelles activitats on l'usuari que està veient el resultat de la vista forma part dels responsables. Per fer-ho afegirem un argument en el bloc "Arguments" que sigui l'identificador de l'usuari de la sessió activa i indicarem que utilitza la relació creada en el bloc anterior.

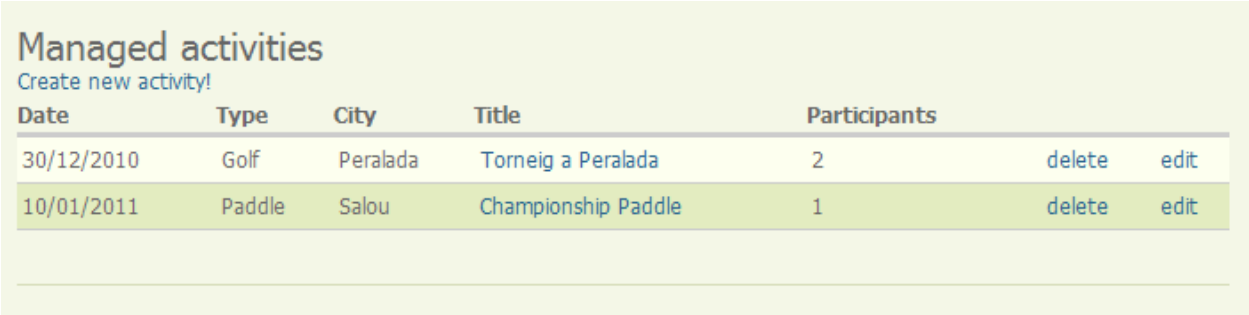


Finalment, en el bloc "Basic settings" s'ha indicat que l'estil de la vista sigui en format de taula. També s'ha decidit crear una paginació de 20 elements per pàgina. Finalment s'ha decidit afegir contingut a la capçalera de la vista en format PHP per poder crear noves activitats. D'aquesta manera damunt la taula amb els resultats veurem un enllaç per crear noves activitat. El codi PHP és el següent:

```php
<?php
print (l('Create new activity!','node/add/activity'));
?>
```

En Drupal la funció "l" ens permet crear enllaços indicant el text de l'enllaç i la URL. En aquest cas, s'ha indicat el path relatiu (des de l'arrel del nostre portal) a la pantalla de creació d'una nova activitat.

Un cop configurada la vista i fetes les proves corresponents el resultat que obtenim és el següent:

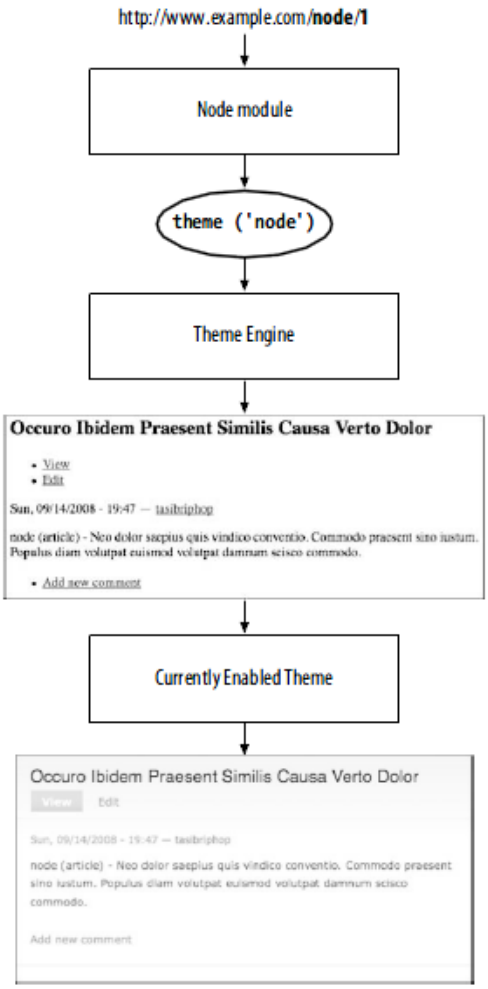


Il·lustració 31 – Vista de les activitats que gestiono

5.2.4 Creació de plantilles per visualitzar activitats, partides i camps de golf

Drupal permet la creació de continguts donant molta llibertat als usuaris per poder afegir tot tipus d'atributs, però a l'hora de mostrar el contingut, ofereix poca llibertat a l'usuari per poder escollir com mostrar el contingut. Bàsicament, la visualització dels nodes depèn d'una plantilla (template) del tema que utilitzem en el portal. Per tant, tots els tipus de continguts tenen per defecte una aparença molt semblant. En canvi, el que el client desitja es que la informació d'una Activitat es vegi de forma diferent a la d'un camp de golf o una partida, on és vol que la informació s'assembli al que podem trobar en una targeta de golf.

Per entendre el funcionament bàsic de com Drupal crea l'HTML de cada contingut adjuntem la següent il·lustració.



Il·lustració 32 – Using Drupal (O'Reilly 2009) – Explicació de com Drupal genera el HTML per a un node

Bàsicament, aquest diagrama ens indica que Drupal, a partir de la URL indicada, primer obté el contingut que s'ha de mostrar. Després la funció "theme('node')" prepara el contingut del node per a què pugui ser interpretat pel "Theme Engine" que a partir dels templates crearà el HTML corresponent i, finalment, el tema aplicarà mitjançant els fitxers CSS el seu estil.

Per interferir en aquest procés, Drupal permet que si en la carpeta on es troben els fitxers amb els templates afegim un fitxer amb la nomenclatura adequada, podem utilitzar aquesta plantilla per visualitzar un tipus de contingut en concret, una vista concreta, etc, de manera que podrem controlar el HTML que es generi.

Finalment, ens hem decantat per una altra opció: utilitzar un mòdul extern anomenat Content Templates[23] que permet la creació i gestió dels templates des del propi portal mitjançant una interfície d'usuari.

Amb Content Templates es poden crear els templates en fitxers i guardar-los en una carpeta indicada pel mòdul ("*sites/all/contemplates/*" partint de la carpeta arrel on tenim el sistema de fitxers del Drupal) o bé guardar-los directament en la base de dades. Un altre punt interessant d'aquest mòdul es que mostra a l'usuari, per cada tipus de contingut, com accedir a la informació de cada atribut del node en PHP. D'aquesta manera ens facilita molt la creació del template al disposar de la nomenclatura exacta de les variables. A continuació mostrem un exemple.

Il·lustració 33 – Exemple de com Content Templates mostra el codi PHP per a cada atribut d'un tipus de contingut

5.2.4.1 Disseny de la funcionalitat

Plantilla Camp de golf

La primera plantilla que crearem serà la del tipus de contingut "Camp de golf". De les 2 opcions disponibles (guardar el template a la base de dades mitjançant un editor html o crear el fitxer del template) escollirem la segona opció ja que ens permet treballar de forma més àgil en un

[23] Més informació del mòdul Content Templates a: http://drupal.org/projects/contentemplates

editor de textos com pot ser Notepad++[24] i alhora ens és molt més ràpid comprovar les modificacions que anem realitzant.

Com s'ha comentat abans, els fitxers es guardaran al directori "sites/all/contemplates". El nom del fitxer ha de tenir el següent format node-tipus_de_contingut-body.tpl.php. La paraula "body" indica que és el template que s'utilitza quan es mostra el node complet. Per tant, en el cas dels camps de golf crearem el fitxer "node-golf_course-body.tpl.php". Sempre han de tenir extensió .tpl.php per indicar que són templates.

Definirem el template en 3 seccions. En la primera secció mostrarem el logotip i la descripció del camp. En la segona, la informació dels forats i, finalment, en la tercera la galeria d'imatges del camp.

El codi HTML+PHP de la primera secció (logotip + descripció) és el següent:

```
<div id="content-content">
    <div style="clear:both;margin-top:10px">
        <div style="float:left;width:250px">
            <?php
                print $node->field_main_photo[0]['view'];
            ?>
        </div>
        <div style="float:right;width:430px">
            <?php
                print $node->content['body']['#value'];
            ?>
        </div>
    </div>
```

Per a mostrar les característiques del camp, crearem una taula indicant les propietats del recorregut.

[24] Més informació de Notepad++ a: http://notepad-plus-plus.org/ (Com a curiositat comentar que el portal de notepad està fen en Drupal utilitzant el mateix tema que porta aquest projecte)

```
<?php $in= array(); $out= array();?>
<fieldset class="fieldgroup"><legend><?php print t('Holes')?></legend>
    <table class="sticky-enabled sticky-table center" style="font-size:90%;">
        <thead>
            <tr>
                <th class="center"></th>
                <?php for($h=1;$h<=9;$h++){?>
                <th class="center"><?php print $course_array['field_number'.$h][0]['view']; ?></th>
                <?php } ?>
                <th class="center"><?php print 'in' ?></th>

                <?php for($h=10;$h<=18;$h++){?>
                <th class="center"><?php print $course_array['field_number'.$h][0]['view']; ?></th>
                <?php } ?>
                <th class="center"><?php print 'out' ?></th>

                </th><th class="center"><?php print 'total' ?></th>
            </tr>
        </thead>
        <tbody>
            <tr class="odd">
                <th class="center"><?php print t('Par')?></th>
                <?php for($h=1;$h<=9;$h++){?>
                <td><?php $in['par']+= $course_array['field_par'.$h][0]['view'];
                          print $course_array['field_par'.$h][0]['view']; ?>
                </td>
                <?php } ?>
            ...
            <tr class="odd">
                <th class="center"><?php print t('HDCP')?></th>
                <?php for($h=1;$h<=9;$h++){?>
                <td><?php print $course_array['field_handicap'.$h][0]['view']; ?></td>
                <?php } ?>
                <td> </td>
                <?php for($h=10;$h<=18;$h++){?>
                <td><?php print $course_array['field_handicap'.$h][0]['view']; ?></td>
                <?php } ?>
                <td> </td>
            </tr>
        </tbody>
    </table>
</fieldset>
```

No s'inclou tota la taula ja que ocuparia massa espai la imatge amb el codi. S'ha eliminat una part on es troben els punts suspensius. En qualsevol cas, la idea és crear una fila per cada atribut i en cada cel·la indicar el valor del forat.

En aquest punt s'ha modificat l'estil per defecte del tema per disposar d'una manera més elegant i vistosa de mostrar el contingut d'una taula HTML. Cada tema té un fitxer d'estil CSS[25] anomenat style.css. En aquest fitxer s'ha afegit el següent codi en CSS:

[25] Més informació a: http://en.wikipedia.org/wiki/Cascading_Style_Sheets

```
/* Score table*/
tr.even td.bold{
  background-color: #D7E2AA;
  font-weight:bold;
}

tr.odd td.bold {
  background-color: #F6F9DB;
  font-weight:bold;
}

tr.even th.bold, tr.odd th.bold {
background-color:#DDDDDD;

}
```

Finalment, es mostra la galeria de fotos. Observar que només hem de mostrar l'atribut ja que el "Theme Engine" s'ha encarregat del HTML de la galeria.

```
<fieldset class="fieldgroup"><legend><?php print t('Images')?></legend>
    <div style="padding-left:20px"><?php print $node->field_gallery[0]['view']?></div>
</fieldset>
</div>
```

Com s'ha pogut veure, el mecanisme de creació del template consisteix en intercalar el codi PHP que mostra el valor dels atributs amb el codi HTML de la plantilla.

Finalment, en la següent il·lustració podem veure el resultat d'aplicar el template en un contingut de tipus "Camp de golf" on es poden veure clarament les 3 seccions.

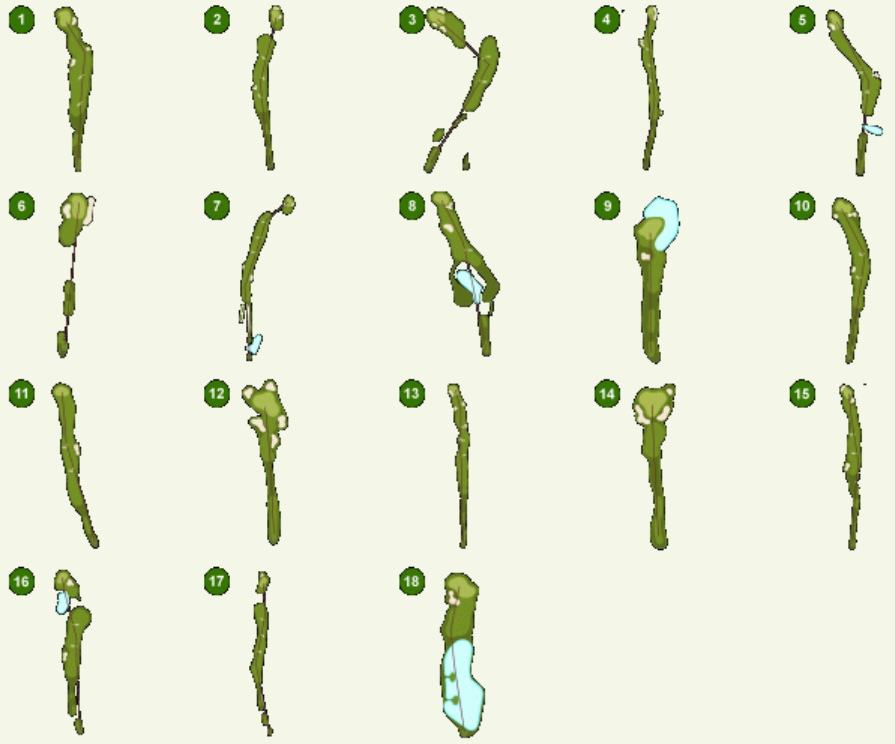
Peralada Golf Club

Magnificent 18-hole course with a distance of 6071 meters, par 71, four tees per hole. With fairly flat areas, lakes, streams and trees, is a field suitable for all types of players.
Fully integrated into the landscape, preserving the environment and caring nature are present in its entirety, the first golf course in Europe to obtain the EMAS Environmental Management Certification recognized by the European Union.
Since its inauguration in 1993, has been recognized as the best golf on the Costa Brava on several occasions and honored by the IAGTO (International Association of Golf Tour Operators), along with other fields of the Association Golf Golf Costa Brava-Girona, the best emerging golf destination in 2000 worldwide.
Competitions and tournaments hosted high-level Open Catalonia in 1995, evidence of Pre-Qualifying School or the PGA Seniors Qualifying School.

Holes

	1	2	3	4	5	6	7	8	9	in	10	11	12	13	14	15	16	17	18	out	total
Par	4	4	4	5	4	3	5	4	3	36	4	4	3	5	3	4	4	5	3	35	71
LG	284	351	358	460	346	186	485	306	160	2936m	284	351	358	460	346	186	485	306	160	2757m	5693m
HDCP	18	2	8	16	6	4	10	14	12		11	9	13	5	17	7	1	3	15		

Images



Il·lustració 35 – Template per visualitzar nodes de tipus “Camp de Golf”

Plantilla Partida de golf

La plantilla per a una partida serà molt semblant a la part de la plantilla que acabem de crear per als camps de golf on es mostra la informació dels forats. La diferencia serà que en aquest cas s'haurà d'incloure la puntuació dels jugadors.

El fitxer que guardarem a la carpeta "sites/all/contemplates" s'anomenarà "node-golf_round-body.tpl.php".

Per obtenir la informació del camp de la partida de golf (en una relació només es guarda l'identificador numèric del node) farem servir de la funció node_load(nid) que donat un identificador numèric et retorna el node amb tots els seus atributs.

```
<?php
$golf_course = node_load($node->field_golf_field[0]['nid'] );
$course_array =get_object_vars($golf_course);
$actual_user=null;
?>
```

A partir d'aquí, per omplir les primeres files de la taula relatives a la informació podem tornar a aprofitar part del template anterior amb la diferencia que indicarem que tota la informació del camp formi part de la capçalera de la taula (de color gris).

Un cop tenim la informació del camp, indicarem la puntuació dels jugadors que han participat en la partida. Només afegirem la fila si el jugador està informat, és a dir, si la partida només te 2 jugadors dels 4 possibles només apareixeran dues files a la taula.

```php
<?php
$node_array =get_object_vars($node);
for($i=0;$i<4;$i++){
        if ($i % 2 == 0) $odd_even='odd';
        else $odd_even='even';
        $in=0; $out=0;
    if ($node->field_players[$i]['uid'] !=0 ) { ?>
    <tr class="<?php print $odd_even?>">
    <?php $player = user_load(array('uid' => $node->field_players[$i]['uid']))?>
        <td><?php print  check_plain($player->name)?></td>
        <?php for($h=1;$h<=9;$h++){$in+=$node_array['field_p'.($i+1).'h'.$h][0]['value'];?>
            <td><?php print $node_array['field_p'.($i+1).'h'.$h][0]['value'];?></td>
        <?php } ?>
            <td class="bold"><?php print $in;?></td>

        <?php for($h=10;$h<=18;$h++){$out+=$node_array['field_p'.($i+1).'h'.$h][0]['value'];?>
            <td><?php print $node_array['field_p'.($i+1).'h'.$h][0]['value'];?></td>
        <?php } ?>
            <td class="bold"><?php print $out;?></td>
            <td class="bold"><?php print $in+$out;?></td>
    </tr>
<?php
}
?>
```

Quan s'afegeixi la funcionalitat d'enregistrar estadístiques de les partides de golf ampliarem la informació d'aquest template per incloure un gràfic amb les estadístiques del jugador.

Llavors, la pantalla per visualitzar partides queda com es mostra en la següent il·lustració:

Round 16/08/2010

Score

Hole	1	2	3	4	5	6	7	8	9	in	10	11	12	13	14	15	16	17	18	out	total
Length	284	351	358	460	346	186	485	306	160	2936m	309	315	151	496	143	362	345	470	166	2757m	5693m
Par	4	4	4	5	4	3	5	4	3	36	4	4	3	5	3	4	4	5	3	35	71
Handicap	18	2	8	16	6	4	10	14	12		11	9	13	5	17	7	1	3	15		
marc	4	5	3	6	4	4	4	6	3	39	2	5	4	6	5	3	5	4	4	38	77
jordi.cabeza	3	3	3	4	3	5	3	3	4	31	3	5	3	3	3	3	4	3	3	30	61
Alex	4	4	4	4	4	4	4	4	4	36	4	4	4	4	4	4	4	4	4	36	72
admin	4	5	4	5	3	3	4	5	6	39	4	2	4	5	6	3	5	4	3	36	75

Il·lustració 36 – Template per visualitzar nodes de tipus "Partida de golf"

Plantilla Activitat

En el cas del tipus de contingut "Activitat" crearem la plantilla no pel fet que no agradi al client la forma del Drupal per visualitzar el contingut, sinó perquè ho necessitarem quan s'hagi d'afegir l'opció de poder apuntar-se a l'activitat sense que s'hagi de fer des de la pantalla d'edició del node. Per aquest motiu, en aquesta secció deixarem preparat el template que modificarem quan s'afegeixi la nova funcionalitat.

El fitxer que guardarem a la carpeta "sites/all/contemplates" s'anomenarà "node-activity-body.tpl.php".

El que farem en aquest punt és copiar el HTML que es crea per defecte al mostrar un contingut de tipus "Activitat" i substituirem els valors dels atributs pel corresponent codi en PHP per mostrar el valor de l'atribut.

A continuació mostrem com a exemple el codi del template on es troba la informació relativa a la direcció on es realitza l'activitat:

```
<fieldset class="fieldgroup group-where"><legend><?php print t('Where')?>: </legend>
    <div class="field field-type-text field-field-address">
        <div class="field-label"><?php print t('Address')?>: </div>
        <div class="field-items">
            <div class="field-item odd"><?php print $node->field_address[0]['safe']?></div>
        </div>
    </div>
    <div class="field field-type-text field-field-city">
        <div class="field-items">
            <div class="field-item odd"><?php print $node->field_city[0]['safe']?></div>
        </div>
    </div>
    <div class="field field-type-text field-field-postcode">
        <div class="field-items">
            <div class="field-item odd"><?php print $node->field_postcode[0]['safe']?></div>
        </div>
    </div>

    <div class="field field-type-content-taxonomy field-field-location">
        <div class="field-label"><?php print t('Location')?>: </div>
        <div class="field-items">
            <div class="field-item odd"><?php print $node->field_location[0]['view']?></div>
        </div>
    </div>
</fieldset>
```

Un detall que hem de tenir en compte és que només haurem de mostrar el camp de golf en cas que l'activitat sigui de tipus golf, en cas contrari no ha d'aparèixer aquesta informació.

```
<?php if($node->field_type[0]['view']=='Golf'){ ?>
<fieldset class="fieldgroup group-golf"><legend><?php print t('Golf')?></legend>
    <div class="field field-type-nodereference field-field-golf-field">
        <div class="field-label"><?php print t('Golf course')?>: </div>
        <div class="field-items">
            <div class="field-item odd"><?php print $node->field_golf_field[0]['view']?></div>
        </div>
    </div>
</fieldset>
<?php } ?>
```

Tot i que no s'ha comentat en els anteriors templates, tots els textos que s'han introduït en cada plantilla (com poden ser les etiquetes dels atributs) es mostren amb la funció t('text a mostrar'). La funció t() de Drupal és la que s'encarrega de traduir text en diferents idiomes. D'aquesta manera si en un futur volem crear el portal en diferents idiomes únicament haurem de definir la traducció en un fitxer específic o mitjançant la interfície gràfica que ens proporcioni Drupal.

En la següent il·lustració podem veure el resultat d'aplicar el template en un contingut de tipus "Activitat".

Can Cuyàs Tournament 18 holes

Where:

Address:
c/ Test s/n
Barcelona
08080
Location:
Barcelona

Golf

Golf course:
Can Cuyàs Golf

When

Start date:
Friday, 18 November, 2011 - 18:00
End date:
Sunday, 20 November, 2011 - 21:00

Who

Responsibles:
jordi.cabeza
Participants:

Description

Tournament in the Can Cuyàs golf course

Images

Il·lustració 37 – Template per visualitzar nodes de tipus "Activitat"

6 Iteració 3

6.1 Llista d'activitats de la iteració

6.1.1 Apuntar-se a les activats

1- Estudi de creació de mòduls en Drupal
2- Estudi de la estructura de la base de dades per saber com es guarda la relació de participants de les activitats
3- Afegir un botó/enllaç en el la pantalla de visualitzar el contingut d'una activitat que permeti apuntar-se a l'activitat
4- Mostrar informació a l'usuari un cop apuntat

6.1.2 Gestió d'estadístiques de les partides de golf

1- Definir quines estadístiques volem guardar
2- Estudiar com crear taules en la base de dades.
3- Creació d'un mòdul que registri les estadístiques de les partides de golf
4- Estudiar les opcions per crear gràfics amb Drupal
5- Crear un gràfic amb les estadístiques en la pantalla de visualització de les partides de golf

6.2 Anàlisi i disseny de les funcionalitats

6.2.1 Apuntar-se a les activats

Fins aquest punt, hem estat treballant amb Drupal amb l'objectiu de treure el màxim profit dels mòduls realitzats per la comunitat de desenvolupadors intentant resoldre les funcionalitats demanades pel client. Però quan es demanen funcionalitats específiques arriba el punt on és més fàcil crear la funcionalitat desitjada que intentar adaptar altres desenvolupaments a les necessitats del projecte. És aquest el cas de la funcionalitat de permetre als usuaris apuntar-se a les "Activitats".

Recordem que en la iteració 1 es van definir els atributs del tipus de contingut "Activitat" i un d'ells era el dels participants, que referenciava els usuaris que volien realitzar l'activitat.

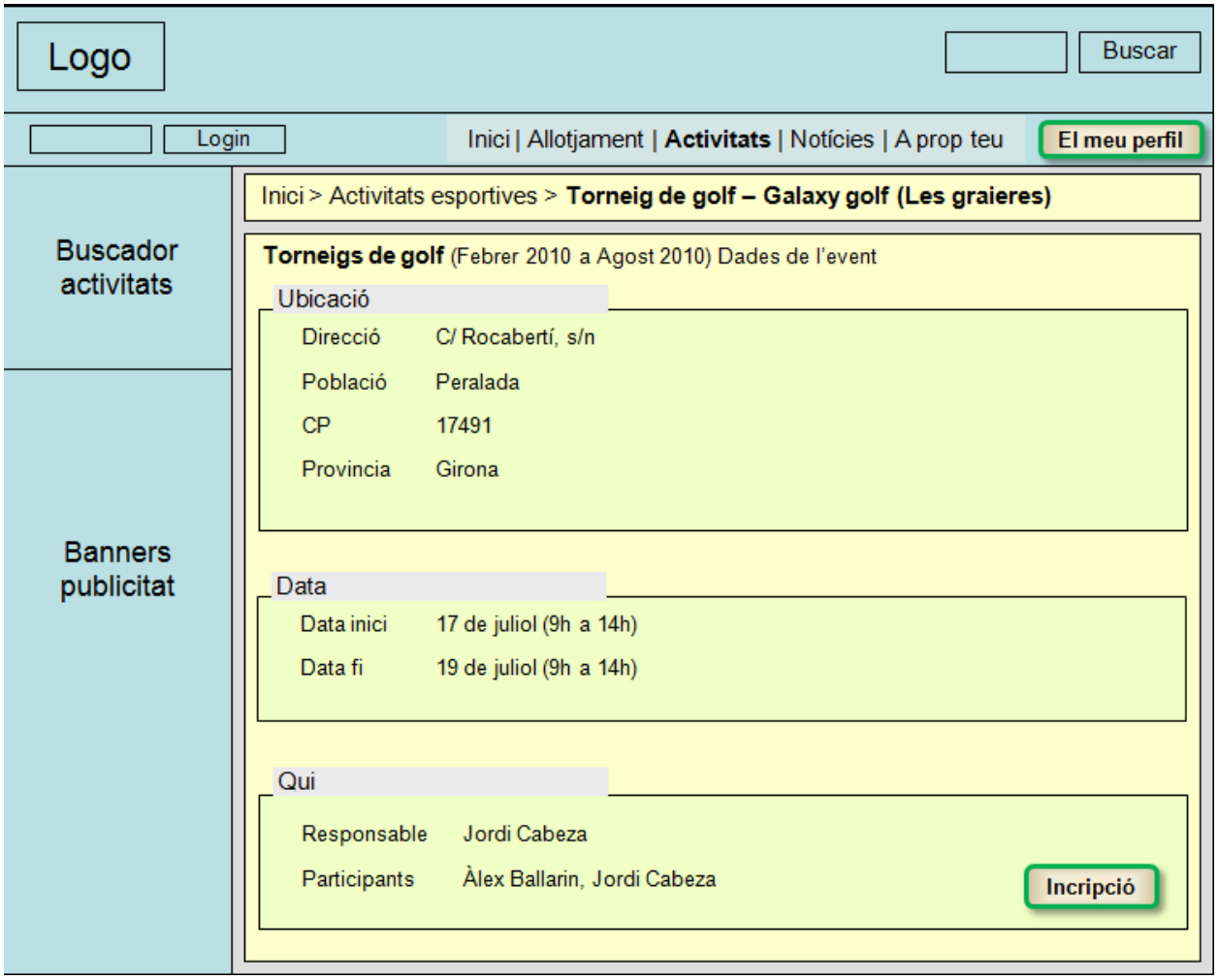
Amb les eines que proporciona Drupal, l'única forma d'apuntar-se a l'activitat és entrant en el formulari d'edició de l'activitat i afegir nous usuaris al camp "Participants" tal com es veu en la següent il·lustració:

Il·lustració 38 – Template per visualitzar nodes de tipus "Activitat"

El problema radica en que cada usuari ha de poder apuntar-se a qualsevol activitat sense haver d'entrar al formulari d'edició. A part, un usuari no ha de poder ni tan sols editar una activitat que no hagi creat prèviament.

Per tant, el que es vol realitzar és permetre en la pantalla de visualització de l'activitat incloure un mecanisme que permeti apuntar-se a les activitats sense haver d'entrar en la pantalla d'edició.

6.2.1.1 Maqueta de la funcionalitat



Il·lustració 39 - Disseny preliminar de la vista de les meves liquidacions

6.2.1.2 Especificació detallada

Del disseny inclòs a l'apartat anterior s'han extret les següent definicions:

1- S'ha d'incloure un botó dins el grup "Qui" en la pantalla de veure una activat que permeti apuntar-se a l'activitat.
2- S'informarà a l'usuari un cop s'hagi realitzat la relació en la base de dades.

6.2.1.3 Disseny de la funcionalitat

Abans de començar a detallar la feina realitzada explicarem el concepte de mòdul i hook en Drupal.

Un mòdul en Drupal és, a grans trets, un conjunt de fitxers organitzats d'una forma determinada i ubicats normalment en la carpeta "sites/all/modules" (partint de l'arrel on tenim instal·lat el Drupal) que realitza una determinada funcionalitat o amplia una de ja existent.

Per crear aquestes funcionalitats Drupal permet la crida de les funcions hooks[26] (ganxo). Aquests hooks són funcions en PHP i es poden considerar com esdeveniments interns de Drupal. Per exemple, amb el ganxo "hook_menu()" podem afegir elements al menú, indicant el seu path i el seu contingut.

Com s'ha explicat en anteriors seccions, en Drupal hi ha 3 tipus de mòduls: els mòduls core (venen ja inclosos de series), els mòduls que afegim a Drupal creats per diferents desenvolupadors per estendre les funcionalitats de Drupal i els mòduls creats per a un desenvolupament projecte en concret (custom modules).

En el sistema de fitxers de Drupal els mòduls core acostumen a ser-hi a la carpeta "/modules". En canvi en una instal·lació habitual de Drupal, els mòduls afegits s'haurien d'ubicar a la carpeta "/sites/all/modules". Finalment, per diferenciar el mòduls que s'han creat específics, aquests s'ubiquen al directori "sites/all/modules/custom". Els mòduls es troben cadascun dintre d'una carpeta amb el seu nom. Per exemple, el mòdul Views es troba a "/sites/all/modules/views".

Un cop s'entén mínimament el funcionament modular de Drupal es pot definir com s'ha realitzar la funcionalitat d'apuntar-se a activitats. Separarem el desenvolupament en 2 parts ben diferenciades: la primera serà la creació del mòdul que permetrà apuntar-se a les activitats i la segona serà afegir el botó a la plantilla d'activitat per apuntar-s'hi.

Creació del mòdul join_activity

[26] Més informació del Hooks de Drupal a: http://api.drupal.org/api/drupal/includes--module.inc/group/hooks/6

Per crear el mòdul el primer pas que realitzarem serà crear la carpeta “join_activity” dins de “sites/all/modules/custom”.

Un cop creada la carpeta, afegirem a la carpeta el fitxer join_activity.info (sempre han de ser de la forma nom_modul.info) que conté la informació del mòdul.

```
; $Id$
name = Join Activity
description = Gives functionality to join an activity without enter in the edit page.
package = PFC Jordi Cabeza
core = 6.x
```

En aquest fitxer s’indica el nom del mòdul, una breu descripció, la versió del Drupal per la qual s’ha dissenyat i el grup de mòduls on s’ubicarà a dins de la pantalla de gestió dels mòduls que ofereix Drupal. S’ha indicat el grup “PFC Jordi Cabeza” on anirem afegint el mòduls creats per el projecte.

Després del fitxer .info, crearem el fitxer amb extensió .module on ubicarem el codi PHP per realitzar la funcionalitat. Els fitxers .module també tenen la nomenclatura nom_modul.module. Per tant, crearem el fitxer join_activity.module.

El primer ganxo (hook) que implementarem serà el hook_menu (que hem definit breument en l’apartat anterior). Amb aquest ganxo definirem una path al qual li associarem una funció, de manera que al ficar en el navegador la ruta executarem la funció que nosaltres desitgem, en aquest cas la que ens permetrà apuntar-nos a les activitats.

```php
/**
* Implementation of hook_menu().
*/
function join_activity_menu() {
    $items['join_activity/join'] = array(
        'page callback' => 'join_activity_join',
        'access callback' => TRUE,
        'type' => MENU_CALLBACK,
    );
    return $items;
}
```

En el codi es pot veure que definim un nou element (ítem) del menú amb el path "join_activity/join" que executarà la funció join_activity_join que definirem a continuació. D'aquesta manera si introduïm al navegador la URL "nom_domini_portal/join_activity/join" Drupal ens mostrarà una pàgina amb el resultat de la funció join_activity_join.

En el mateix fitxer .module definirem la funció. A continuació es detalla el codi PHP acompanyat de la explicació del que es realitza a cada pas.

```php
function join_activity_join($activity_nid = '', $activity_vid='', $user_id = '') {
    //Comprovar si existeix relació
    $sql="SELECT count(*) FROM {content_field_participantes}
          WHERE nid = %d AND vid = %d AND field_participantes_uid= %d";
    $exist= db_result(db_query($sql, $activity_nid, $activity_vid, $user_id));

    if($exist==0) //No existeix la relació
    {
        $sql="SELECT max(delta) FROM {content_field_participantes} WHERE nid = %d AND vid = %d";
        $delta= db_result(db_query($sql, $activity_nid, $activity_vid));

        $sql="INSERT INTO {content_field_participantes} (nid,vid,delta,field_participantes_uid)
              VALUES (%d, %d, %d, %d)";
        db_query($sql, $activity_nid, $activity_vid, $delta+1, $user_id);

        //delete node from cache
        db_query("DELETE FROM cache_content WHERE cid = '%s'",'content:'.$activity_nid.':'.$activity_vid);

        $result = '<div class="messages status">'.t('Joined to this activity.').'</div>';
    }else{ //Existeix la relació
        $result = '<div class="messages error">'.t('You have already joined to this activity.').'</div>';
    }
    return $result;
}
```

Seguint la numeració:

1. Primer es comprovarà si l'usuari que es vol apuntar a l'activitat ja s'ha apuntat prèviament. Per fer-ho s'ha de conèixer com Drupal enregistra aquesta informació em la base de dades. Després d'investigar l'estructura descobrim que per cada atribut de tipus referència (a un usuari o un node) Drupal, més concretament el mòdul CCK, crea una taula amb la següent informació:

- Versió del node de la relació (per defecte té el valor de l'identificador del node si no s'ha creat una nova versió del node)
- Identificador numèric del node (en aquest cas el identificador de l'activitat)
- Número per indicar un ordre (delta). Per exemple, en el cas d'apuntar-se a activitats l'usuari que primer s'apunti tindrà un valor 0, el segon un 1, el tercer un 2, etc.
- Identificador numèric del node o usuari amb el que es relaciona (en aquest cas, l'usuari que s'apunta a l'activitat).

Campo	Tipo	Cotejamiento	Atributos	Nulo
vid	int(10)		UNSIGNED	No
nid	int(10)		UNSIGNED	No
delta	int(10)		UNSIGNED	No
field_participantes_uid	int(10)		UNSIGNED	Sí

Il·lustració 40 – Camps de la taula de la base de dades "content_field_participantes"

En aquest cas, la taula s'anomena "content_field_participantes". Llavors, per saber si l'usuari està apuntat necessitem saber el "vid" i "nid" del node i l'identificador de l'usuari. Justament a la capçalera de la funció tenim tres paràmetres amb aquests valors. Per indicar els paràmetres quan es fa la crida s'afegeixen un a un al path que s'ha definit separat per "/". En el nostre cas seria "/join_activity/join/valor_param1/valor_param2/valor_param3".

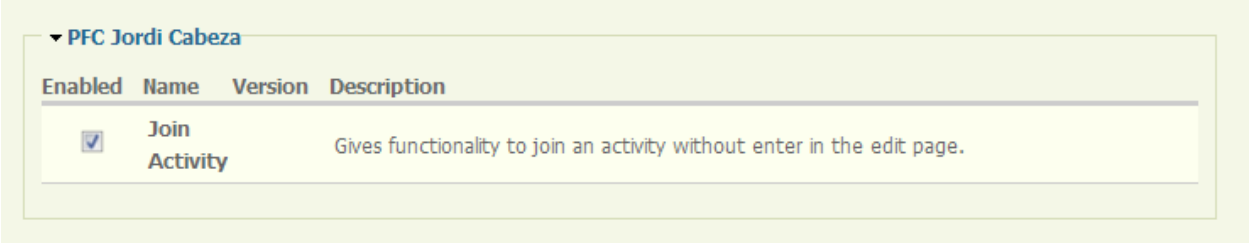
Llavors, ara només queda realitzar la consulta a la base de dades utilitzant les funcions db_query() i db_result() per crear i obtenir el resultat de la consulta respectivament.

2. En aquest segon bloc, que s'executarà quan l'usuari encara no participa, afegirem la relació en la base de dades i es retornarà un missatge a l'usuari indicant que s'ha apuntat a l'activitat
3. El tercer bloc només s'executarà quan l'usuari ja està apuntat i retornarà un missatge que ho indiqui.

Al provar el funcionament de la funció es va trobar que es creava de forma correcta la relació a la base de dades però al visualitzar l'activitat sortien els mateixos participants que abans.

Finalment, es va descobrir que com que no s'havia editat el node mitjançant el formulari d'edició, Drupal no s'assabentava del canvi i retornava la informació que tenia del node a la cache (que corresponia a l'últim cop que s'havia editat el node). Per tant, es va investigar com es podia eliminar el node de la cache. Per fer-ho simplement s'ha hagut d'esborrar el registre de la taula "cache_content" tal com es mostra en el codi en la imatge anterior.

Per acabar aquest punt activarem el mòdul en la pantalla d'administració de Drupal (en la pàgina "http://nom_portal/admin/build/modules").



Il·lustració 41 – Pantalla de configuració de mòduls. Mòdul "join_activity"

Afegir botó per apuntar-se a una activitat

Un cop tenim el mòdul creat, tornarem al template de l'activitat creat en la iteració anterior per afegit un botó que ens permeti apuntar-nos a l'activitat.

Si el botó l'implementem com un enllaç a la ruta definida en el mòdul per executar la funcionalitat d'apuntar-se, Drupal ens retornarà una pàgina on veurem el missatge que indicarà si l'usuari s'ha apuntat o ja ho estava. Això no interessa ja que se surt de la pàgina on estem (veient una activitat) . L'opció desitjada es que aquesta funcionalitat es pugui realitzar sense haver de sortir de la pantalla.

Per aconseguir aquest comportament, ens ajudarem de jQuery[27], una llibreria de javascript molt estesa que es pot integrar fàcilment a Drupal. De fet, ja venia integrada degut al

[27] Més informació de jQuery a: http://jquery.com/

desenvolupament previ. Amb jQuery farem la crida a la funció, ens quedarem amb el HTML que conté el missatge que es mostra a l'usuari i el mostrarem per pantalla.

```
$(function() {
    $.get(
        '/drupalPFC/join_activity/join/<?php print $node->nid.'/'.$node->vid.'/'.$GLOBALS['user']->uid?>',
        function(data) {$('#join_activity').html($(data).find('#content-content'));}
    );
});
```

El que farem és utilitzar la funció get(URL, function()) que permet cridar a una URL i executar una funció on poder manipular el resultat. En aquest cas es crida la ruta "join_activity/join/" indicant els paràmetres identificador del node, de la seva versió i de l'identificador de l'usuari (l'obtenim de la variable global on es guarda la informació de l'usuari que té sessió en el sistema). En la funció per manipular el resultat obtenim el missatge a l'usuari i el mostrem.

Ara només queda afegir un botó a la plantilla que executi aquest codi quan sigui clicat.

```
<div id="join_activity">
    <span style="float:right;">
        <input class="teaser-button" type="submit" value="Join it!"
            onclick="$(function() {$.get('/drupalPFC/join_activity/join/...>
    </span>
</div>
```

Finalment només queda veure el funcionament del mòdul. En la parta esquerra mostra quan l'usuari s'apunta a una activitat on no estava apuntat i rep el missatge de confirmació. En canvi, en la parti dreta, l'usuari s'intenta apuntar a una activitat on ja s'ha apuntat previament i rep el missatge d'error corresponent.

Il·lustració 42 – Funcionament del mòdul "join_activity"

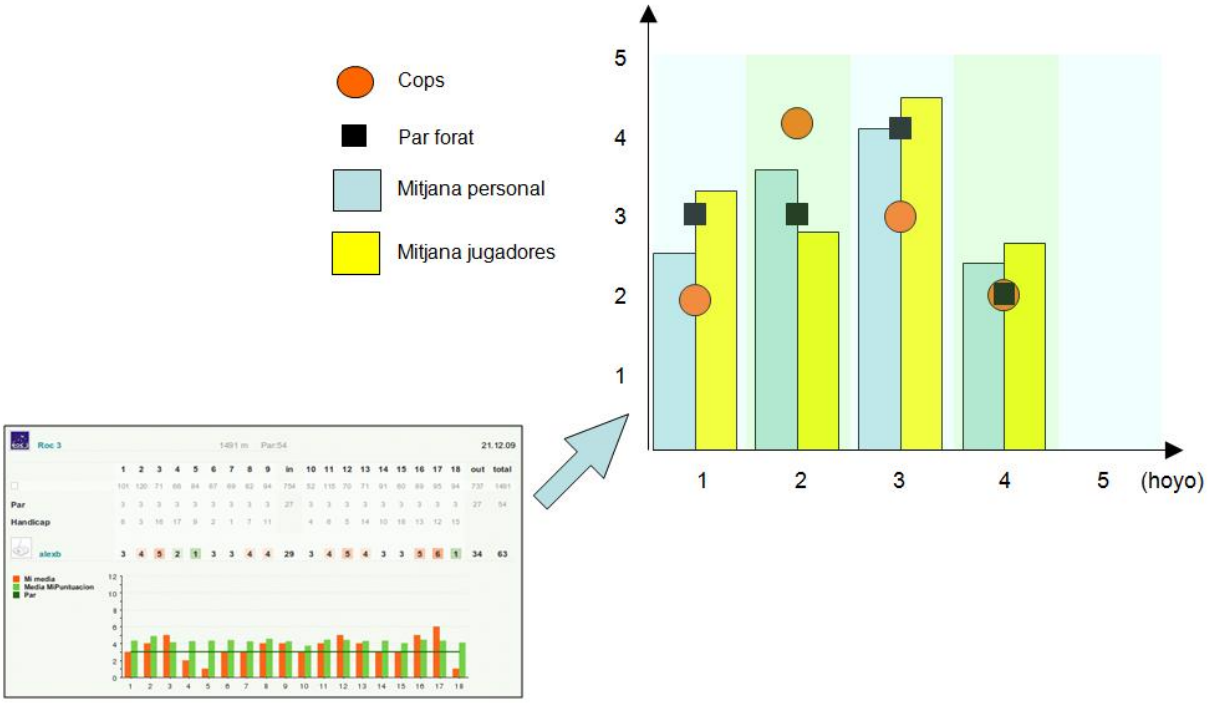
6.2.2 Gestió d'estadístiques de les partides de golf

En aquest punt del projecte, on ja tenim un coneixement força avançat de Drupal, s'haurà de crear un mecanisme que permeti mostrar a l'usuari en una gràfica les següents estadístiques de la partida:

- Els cops que ha realitzat el jugador en cada forat
- El par de cada forat del camp
- La seva mitjana de cops per a cada forat del camp
- La mitjana de tots els jugadors en cada forat del camp

D'aquesta manera podrem veure gràficament si el resultat del jugador a cada forat està per sota o per sobre del par, si està per sota o per sobre de la seva mitjana i si té un millor o pitjor resultat del que acostuma a realitzar la resta de jugadors. També es veurà de forma gràfica si el resultat global de la partida millora o empitjora respecte al seu històric o l'històric global dels jugadors.

6.2.2.1 Maqueta de la funcionalitat



Il·lustració 43 – Maqueta de la funcionalitat per obtenir estadístiques de les partides de golf

6.2.2.2 Especificació detallada

Del disseny inclòs a l'apartat anterior s'han extret les següent definicions:

1. S'ha de crear un diagrama múltiple on l'eix Y serà el número de cops i l'eix X el número dels forats que contindrà la següent informació:
    - Diagrama de punts amb els cops de l'usuari a cada forat. Si es pot s'uniran els punts amb una línia (sense perdre la visualització dels punts).
    - Diagrama de punts amb el par de cada forat. Si es pot s'uniran els punts amb una línia (sense perdre la visualització dels punts). Ha de ser en un color diferent al de cops de l'usuari.
    - Diagrama de barres amb la mitjana de cops per forat de l'usuari en el camp representat amb un color diferent a la resta.

- Diagrama de barres amb la mitjana dels jugadors per forat en el camp representat amb un color diferent a la resta.

2- S'ha d'afegir el diagrama en el template que mostra les estadístiques just a sota de la puntuació de la partida. Només es mostrarà si l'usuari de la sessió del sistema ha participat en la partida.

6.2.2.3 Disseny de la funcionalitat

La primera tasca que s'ha de realitzar és decidir com obtindrem la informació de les estadístiques. Hi ha dues opcions, la primera és calcular la informació de les partides quan es mostra la puntuació d'una partida i la segona és guardar/modificar les estadístiques en la base de dates cada cop que es crea, es modifica o es borra una partida.

Com que serà més freqüent l'opció de consulta que l'opció de crear/modificar/esborrar partides escollirem la segona opció. No tindria sentit haver de calcular l'històric de tots els jugadors, o el d'un jugador, cada cop que es visualitza una partida.

Per tant, s'ha de definir l'estructura de les taules que crearem a la base de dades. Per guardar a la base de dades les estadístiques crearem les 2 taules següents:

golf_stats_course			
Fields			
Field	**Type**	**Null**	**Key**
cid	int(11)	NO	PRI
rounds	int(11)	NO	
hole1	float unsigned	NO	
hole2	float unsigned	NO	
hole3	float unsigned	NO	
hole4	float unsigned	NO	
hole5	float unsigned	NO	
hole6	float unsigned	NO	
hole7	float unsigned	NO	
hole8	float unsigned	NO	
hole9	float unsigned	NO	
hole10	float unsigned	NO	
hole11	float unsigned	NO	
hole12	float unsigned	NO	
hole13	float unsigned	NO	
hole14	float unsigned	NO	
hole15	float unsigned	NO	
hole16	float unsigned	NO	
hole17	float unsigned	NO	
hole18	float unsigned	NO	

golf_stats_user_course			
Fields			
Field	**Type**	**Null**	**Key**
uid	int(11)	NO	PRI
cid	int(11)	NO	PRI
rounds	int(11)	NO	
hole1	float unsigned	NO	
hole2	float unsigned	NO	
hole3	float unsigned	NO	
hole4	float unsigned	NO	
hole5	float unsigned	NO	
hole6	float unsigned	NO	
hole7	float unsigned	NO	
hole8	float unsigned	NO	
hole9	float unsigned	NO	
hole10	float unsigned	NO	
hole11	float unsigned	NO	
hole12	float unsigned	NO	
hole13	float unsigned	NO	
hole14	float unsigned	NO	
hole15	float unsigned	NO	
hole16	float unsigned	NO	
hole17	float unsigned	NO	
hole18	float unsigned	NO	

Il·lustració 44 – Estructura de les taules per guardar estadístiques de les partides de golf

La taula "golf_stats_course" guardarà la mitjana de cops dels usuaris per a cada forat d'un camp i el número de partides jugades. En canvi, la taula "golf_stats_user_course" guardarà la mitjana de cops per forat d'un usuari en un camp determinat.

Amb les taules definides, dividirem la feina en dos tasques: la primera crear el mòdul que guardi les estadístiques i la segona.

Creació del mòdul golf_stats

Crearem en primer terme la carpeta "golf_stats" dins de "sites/all/modules/custom" on allotjarem els fitxers del mòdul.

Amb la carpeta creada crearem el fitxer "golf_stats.info" amb la informació bàsica del mòdul.

```
; $Id$
name = Golf Stats
description = Record stats of user rounds.
package = PFC Jordi Cabeza
core = 6.x
```

En el punt anterior hem definit com han de ser les taules de la base de dades on guardarem les estadístiques, tot i que no les hem creat. Aquestes taules només tenen sentit en la base de dades si el mòdul "golf_stats" es troba actiu. En cas contrari serien taules inútils.

Drupal permet definir en un mòdul les accions a realitzar quan aquest s'està instal·lant o eliminant del sistema mitjançant el ganxos hook_install() i hook_uninstall(). Perquè Drupal executi aquest ganxos s'han d'incorporar en un fitxer ".install" en el mòdul. Per tant, crearem el fitxer "golf_stats.install"

```
/**
 * Implementation of hook_install().
 */
function golf_stats_install() {
  drupal_install_schema('golf_stats');


  drupal_set_message('Golf Stats installed.');
}

/**
 * Implementation of hook_uninstall().
 */
function golf_stats_uninstall() {
  drupal_uninstall_schema('golf_stats');

  // Delete all the conditional fields variables and then clear the variable cache
  db_query("DELETE FROM {variable} WHERE name LIKE 'golf_stats_%'");
  cache_clear_all('variables', 'cache');

}
```

Quan instal·lem els mòduls afegirem l'esquema "golf_stats" a la base de dades i al desinstal·lar l'esborrarem. A més a més, esborrarem les possibles variables que s'hagin pogut crear del mòdul en el sistema. Per definir les taules utilitzarem el ganxo hook_schema(). A continuació mostrem part del codi per fer-nos una idea de com es defineixen les taules.

```php
function golf_stats_schema() {
  $schema = array();
  $schema['golf_stats_user_course'] = array(
    'description' => 'Stores golf stats of user in one course.',
    'fields' => array(
        'uid' => array(
            'type' => 'int',
            'length' => 10,
            'not null' => TRUE,
            'default' => 0,
            'description' => 'User id.',
        ),
        'cid' => array(
            'type' => 'int',
            'length' => 10,
            'not null' => TRUE,
            'default' => 0,
            'description' => 'The node id of the course.',
        ),
        'rounds' => array(
            'type' => 'int',
            'length' => 10,
            'not null' => TRUE,
            'default' => 0,
            'description' => 'Number of rounds played at the course with fid by user with uid.',
        ),
```

Com es pot veure, les taules no es defineixen en SQL sinó que es fan mitjançant "arrays" de PHP. Això permet haver de preocupar-se si la definició serà compatible amb el gestor de bases de dades utilitzats, ja que Drupal s'encarregarà de muntar la comanda SQL corresponent a cada gestor.

Amb tota aquesta feina prèvia realitzada, ja es pot crear la funcionalitat en el fitxer "golf_stats.module". El codi PHP que crearem ha de satisfer les següents necessitats:

1- En el cas de crear una nova partida s'han d'actualitzar les estadístiques de cada jugador en el camp de la partida en la taula "golf_stats_user_course". Utilitzant la següent funció per calcular la mitjana actualitzada de cada forat:

$$mitjanaForatX_{nova} = \frac{(mitjanaForatX_{antiga} \times numeroPartides) + puntuacioForatX}{(numeroPartides + 1)}$$

Aplicarem el mateix càlcul per a la taula "golf_stats_user_course" per calcular la mitjana de tots els jugadors. La diferència és que en aquesta taula només hi ha un registre per camp de manera que al afegir una nova partida actualitzem aquesta mitjana tants cops

com jugadors han participat. En les dues taules augmentaren el número de partides jugades de forma corresponent.

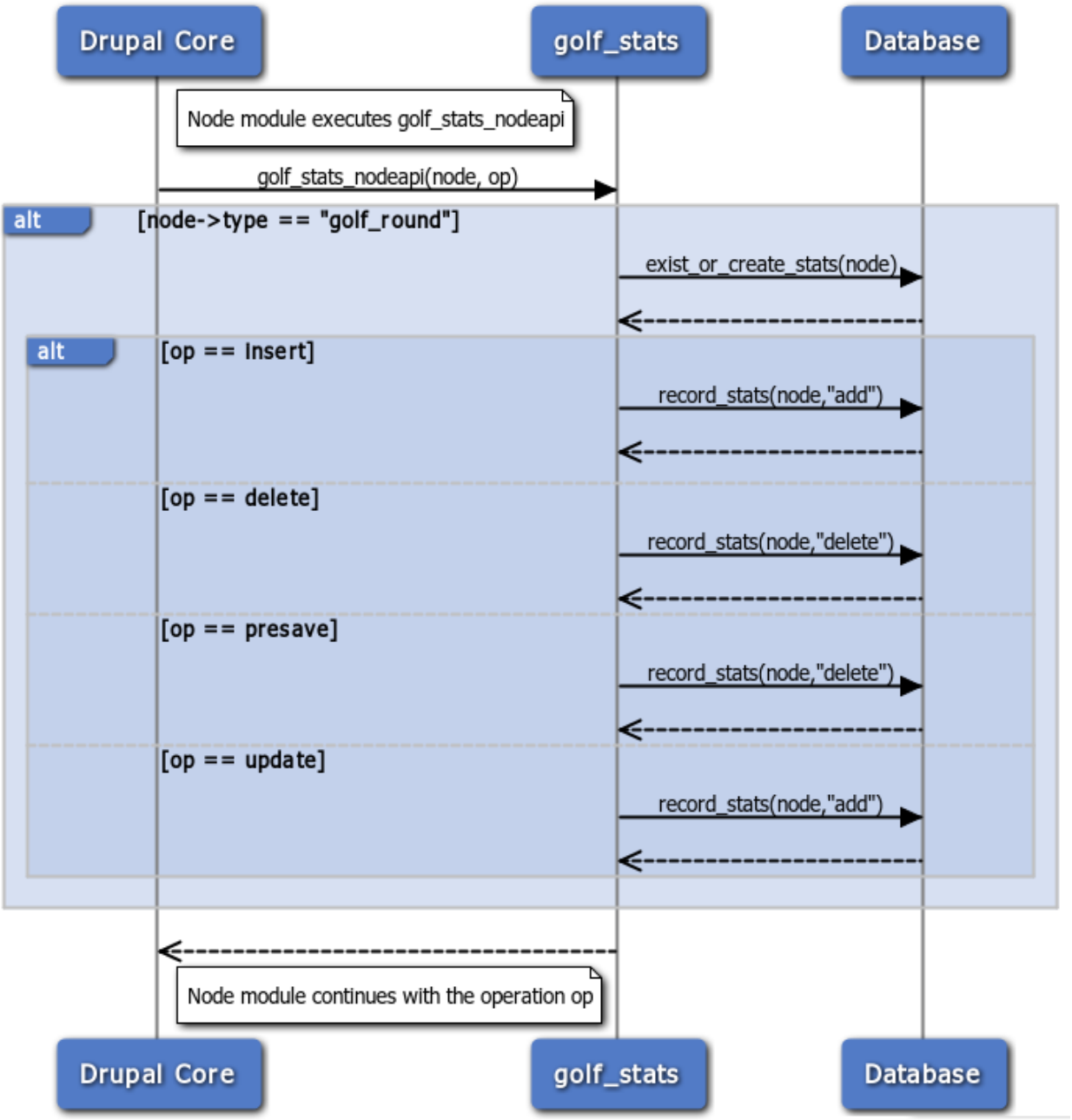
2- Quan s'esborri una partida existent s'han d'actualitzar les estadístiques de cada jugador en el camp de la partida en la taula "golf_stats_user_course". Utilitzarem la següent funció per calcular la mitjana actualitzada de cada forat:

$$mitjanaForatX_{nova} = \frac{(mitjanaForatX_{antiga} \times numeroPartides) - puntuacioForatX}{(numeroPartides - 1)}$$

Aplicarem el mateix càlcul per a la taula "golf_stats_user_course".

3- Finalment, en el cas d'actualitzar una partida farem una barreja dels 2 casos anteriors. Primer esborrarem la partida amb la puntuació sense modificar i després l'afegirem amb la nova puntuació.

Per poder crear aquest funcionament ens ajudarem del ganxo hook_nodeapi(). Aquest ganxo ens permet agafar el control de l'execució just en el moment de realitzar una operació sobre el node. D'aquesta manera podem interrompre l'execució de l'operació, actualitzar les estadístiques si el node es de tipus "Partida de golf" i tornar a donar el control de l'execució. El següent diagrama mostra el comportament del ganxo golf_stats_nodeapi() que implementarem:



A continuació mostrem el codi PHP que implementa el funcionament descrit en el diagrama:

```php
/**
 * @file
 * Record stats of user
 */
function golf_stats_nodeapi(&$node, $op, $a3 = NULL, $a4 = NULL) {

    if ($node->type == "golf_round") {
        exist_or_create_stats($node);
        switch ($op) {
            case "presave":
                if ($node->nid!=0){ //Si existeix el node
                    $node_DB = node_load($node->nid);
                    record_stats($node_DB,"delete");
                }
                break;
            case "insert":
                record_stats($node,"add");
                break;
            case "update":
                record_stats($node,"add");
                break;
            case "delete":
                record_stats($node,"delete");
                break;
        }
    }
}
```

S'ha comentat abans que quan es modifica una partida primer s'esborren les estadístiques. Això ho fem quan l'operació és igual a "presave" on podem agafar el control després d'enviar les dades per actualitzar però abans que s'hagin desat a la base dades. Així podem accedir a la puntuació antiga i esborrar-la per després afegir la nova puntuació com si fos una nova partida.

La funció "exist_or_create_stat(node)" mira si s'estan guardant estadístiques del camp i dels jugadors. En cas que no es guardin, es crearà un registre a la base de dades amb mitjana de 0 cops per forat i 0 partides jugades.

La funció record_stats(node, op) esborra les estadístiques de la partida si el paràmetre "op" és igual a "delete" o les afegeix si és igual a "add".

No s'inclou el codi d'aquestes funcions ja que ocuparien massa espai al document i no són gaire significatives.

El gran avantatge de tot aquest procés es que resulta transparent a l'usuari ja que no ha de realitzar cap acció extra perquè es guardin les estadístiques.

Finalment, activarem el mòdul en la pantalla d'administració de mòduls ("admin/build/modules").

Després de fer les proves corresponents de la funcionalitat ens hem adonat que perquè el mòdul funcioni correctament s'han d'esborrar totes les partides creades fins al moment, ja que aquestes no tenen estadístiques. En principi, quan el portal es posi en funcionament no disposarà de partides creades i el mòdul "golf_stats" es trobarà actiu. D'aquesta manera les estadístiques es guardaran correctament. En cas que es volguessin enregistrar les estadístiques de les partides creades abans d'activar el mòdul s'hauria de crear una funció que s'executés al instal·lar (el mòdul) i que enregistrés les estadístiques de les partides ja creades.

Creació de diagrames que mostrin les estadístiques

Amb el mòdul creat, tornarem al template per visualitzar partides de golf creat en la iteració anterior per afegir els diagrames que mostrin les estadístiques.

Per crear diagrames a Drupal podem trobar diferents alternatives com pot ser utilitzar la api de Google Charts o integrar altres llibreries per crear gràfiques. Després de realitzar diferents proves s'ha decidit utilitzar el mòdul Open Flash Chart API[28]. Aquest mòdul permet la creació de diagrames en PHP utilitzant la llibreria amb el mateix nom que el mòdul. S'ha decidit utilitzar aquesta opció ja que permet crear diagrames vistosos a l'usuari de forma relativament fàcil gràcies a la documentació que trobem a la pàgina web de l'autor[29].

[28] Més informació a: http://drupal.org/project/open_flash_chart_api
[29] Més informació de la llibreria Open Flash Chart a: http://teethgrinder.co.uk/open-flash-chart/

L'únic inconvenient que trobem per utilitzar aquesta llibreria és haver de disposar del plugin de Flash instal·lat en el navegador. No obstant, assumim aquest inconvenient ja que gairebé tots els usuaris disposen de flash al seu navegador.

Un cop escollida la tecnologia a utilitzar per crear el diagrama ens dirigirem a la plantilla de "Partides de golf" (fitxer "sites\all\contemplates\node-golf_round-body.tpl.php")per afegir el codi necessari per mostrar-ho. Recordarem que el diagrama només es mostrarà si l'usuari que està visualitzant la partida ha participat en ella com a jugador. Per tant, el primer pas és mirar si l'usuari actual participa o no a la partida. En cas afirmatiu guardarem la seva puntuació, el par del camp, l'històric del jugador i l'històric del camp.

```
//puntuació de l'usuari
$data_1 = $user_score;

//par camp de golf
$data_2 = array($golf_course->field_par1[0]['value'],$golf_course->field_par2[0]['value'], ...

//històric usuari
$sql="SELECT * FROM {golf_stats_user_course} WHERE uid = %d AND cid = %d";
$result= db_query($sql, $actual_user->uid, $node->field_golf_field[0]['nid']);
$data_3 = array();
while ($user_stats = db_fetch_array($result)) {
    for($i=1;$i<=18;$i++){
        $data_3[$i]=$user_stats['hole'.$i];
    }
}

// històric jugadors
$sql="SELECT * FROM {golf_stats_course} WHERE cid = %d";
$result= db_query($sql,$node->field_golf_field[0]['nid']);
$data_4 = array();
while ($user_stats = db_fetch_array($result)) {
    for($i=1;$i<=18;$i++){
        $data_4[$i]=$user_stats['hole'.$i];
    }
}
```

Amb la informació preparada ja podem crear el diagrama en la plantilla.

```
$g = new open_flash_chart_api();
$g->bg_colour = '#F4F7E7';
$g->set_title( $golf_course->title, '{font-size: 20px; color: #666666;margin:10px;}'
$g->set_width(686);
$g->set_height(250);
$g->set_num_decimals( 1 );
$g->set_is_decimal_separator_comma( true );

$g->set_tool_tip( t('Hole').': #x_label#<br>'.t('Hits').': #val#' );

$g->set_data( $data_1 );
$g->line_dot( 3, 5, '#FFCD19', t('This round'), 11 );    // <-- 3px thick + dots

$g->set_data( $data_2 );
$g->line_dot( 3, 5, '#666666', t('Par'), 11 );    // <-- 3px thick + dots

$g->set_data( $data_3 );
$g->bar_glass( 50, '#2C74A3','#1F4C6B' ,t('Average').' '.$actual_user->name, 11 );

$g->set_data( $data_4 );
$g->bar_glass( 50, '#94CE18', '#678E11',t('Average').' ' .t('players'), 11 );


$g->set_x_labels( array( '1','2','3','4','5','6','7','8','9',
                         '10','11','12','13','14','15','16','17','18' ) );
$g->set_y_max( 8 );
$g->y_label_steps( 4 );

$g->set_y_legend( t('Hits'), 12, '#666666' );
$g->set_x_legend( t('Hole'), 12, '#666666' );

$g->x_axis_colour( '#666666', '#808080' );
$g->y_axis_colour( '#666666', '#808080' );

echo '<fieldset class="fieldgroup"><legend>'.t('Stats').'</legend>';
echo $g->render();
echo '</fieldset>';
```

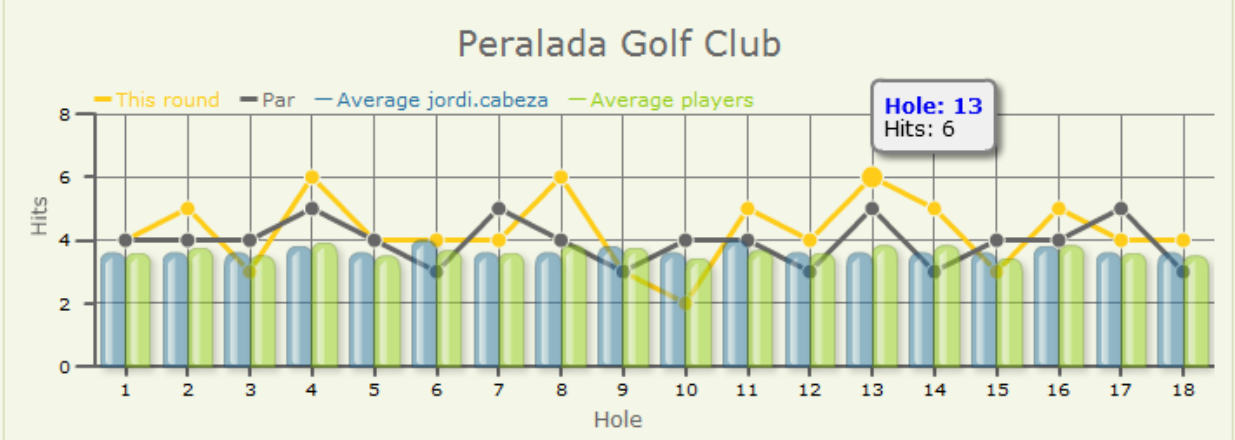
Un cop afegit el codi a la plantilla arriba l'hora de provar que tot funciona correctament. En la següent il·lustració podem veure com queda la pantalla de visualització d'una partida amb el diagrama per mostrar estadístiques.

Round 16/08/2010

Score

Hole	1	2	3	4	5	6	7	8	9	in	10	11	12	13	14	15	16	17	18	out	total
Length	284	351	358	460	346	186	485	306	160	2936m	309	315	151	496	143	362	345	470	166	2757m	5693m
Par	4	4	4	5	4	3	5	4	3	36	4	4	3	5	3	4	4	5	3	35	71
Handicap	18	2	8	16	6	4	10	14	12		11	9	13	5	17	7	1	3	15		
marc	4	5	3	6	4	4	4	6	3	39	2	5	4	6	5	3	5	4	4	38	77
jordi.cabeza	3	3	3	4	3	5	3	3	4	31	3	5	3	3	3	3	4	3	3	30	61
Alex	4	4	4	4	4	4	4	4	4	36	4	4	4	4	4	4	4	4	4	36	72
admin	4	5	4	5	3	3	4	5	6	39	4	2	4	5	6	3	5	4	3	36	75

Stats



Il·lustració 45 – Visualització d'una partida de golf amb estadístiques

7 Iteració 4

7.1 Llista d'activitats de la iteració

7.1.1 Vista de les partides de golf de l'usuari

1- Crear una vista que permeti veure als usuaris el llistat de les partides de golf que han disputat agrupades pel camp de golf.
2- Permetre mostrar la puntuació de la partida de golf sense haver de sortir de la pantalla amb el resultat de la vista.
3- Permetre filtrar les partides de golf segons el camp de golf on s'han realitzat.

7.1.2 Creació del perfil d'usuari

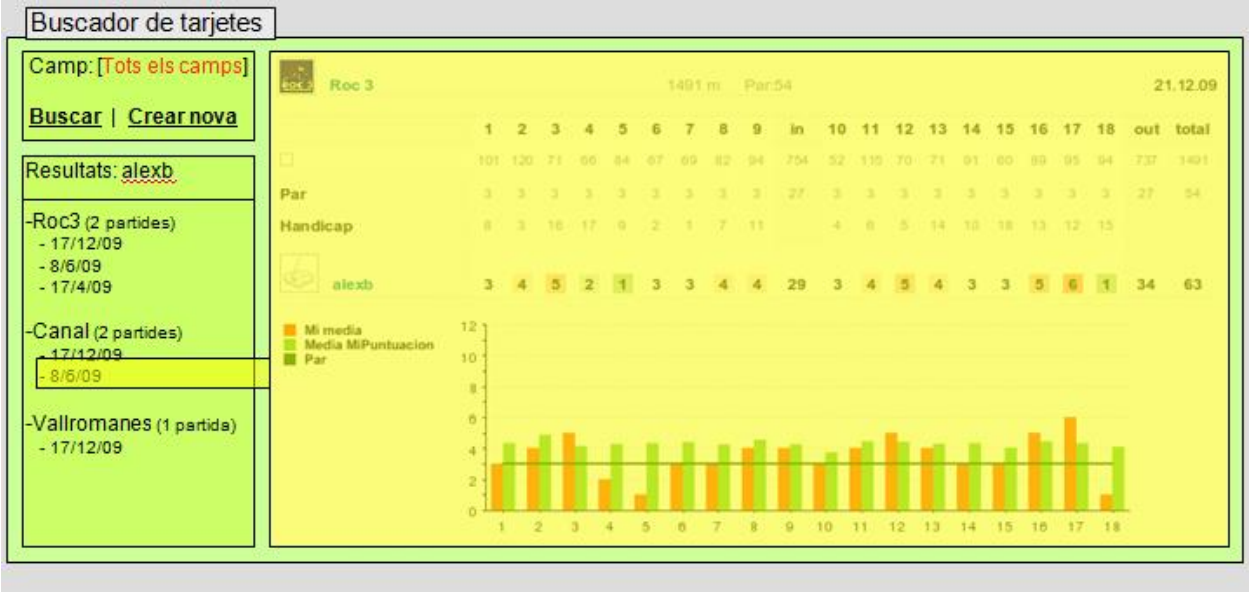
1- Estudiar com crear una pàgina que pugui contenir diferents blocs de continguts separats gràficament.
2- Afegir la informació de l'usuari relacionada amb les activitats (activitats on participa i activitats que gestiona) com un bloc de contingut.
3- Afegir la vista de les partides de golf de l'usuari com un bloc de contingut.
4- Permetre afegir 2 blocs de contingut relacionats amb la informació personal de l'usuari i la informació relacionada amb el lloguer de cases rurals. Aquest 2 blocs formen part del desenvolupament paral·lel a aquest projecte.

7.2 Anàlisi i disseny de les funcionalitats

7.2.1 Vista de les partides de golf de l'usuari

Amb aquesta vista es pretén que l'usuari pugui disposar d'un llistat de les partides de golf realitzades per poder consultar fàcilment les puntuacions realitzades a cada partida.

7.2.1.1 Maqueta de la funcionalitat



Il·lustració 46 – Maqueta de la vista de partides de golf de l'usuari

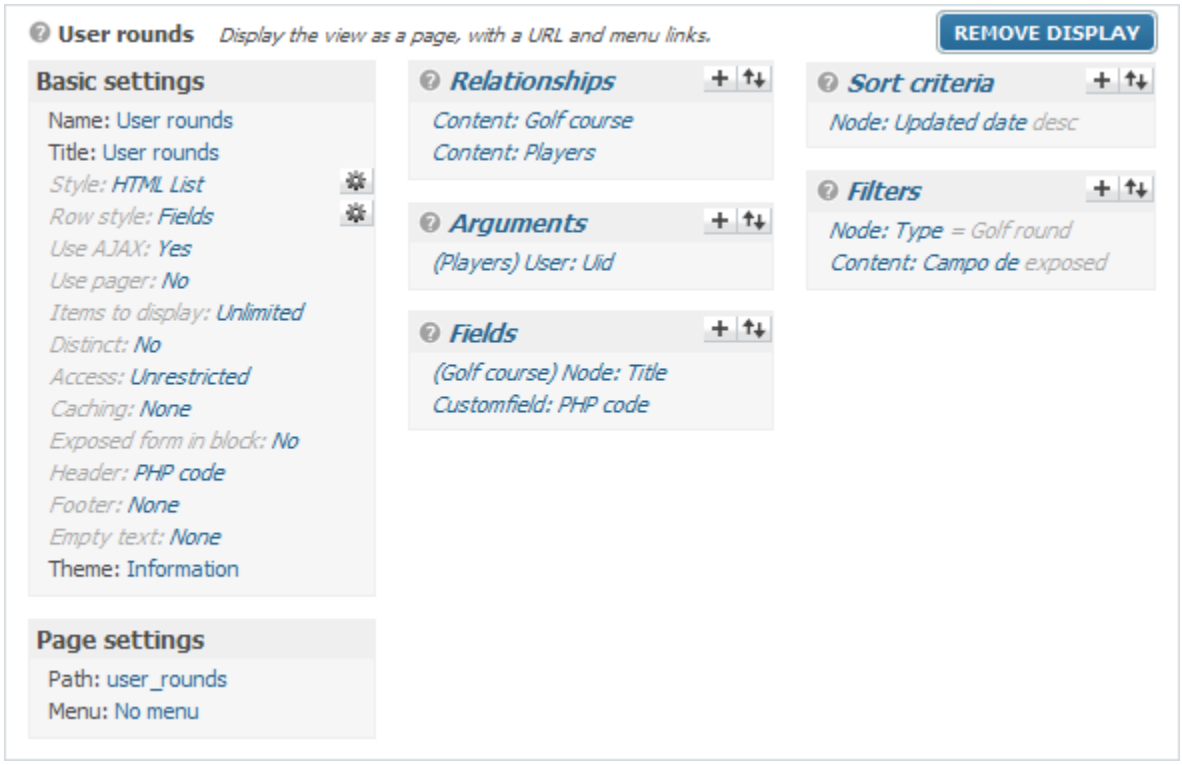
7.2.1.2Especificació detallada

Del disseny inclòs i de la informació extreta en les diferents reunions amb el client obtenim que la vista ha de complir les següents necessitats:

- S'han de mostrar les partides en un llistat agrupades per camp de golf ordenades per data de més nova a més antiga
- S'ha de permetre filtrar per camp de golf
- S'ha d'incorporar un enllaç per què els usuaris puguin crear noves partides amb facilitat
- Al clicar en una partida del llistat s'ha de mostrar el resultat de la partida sense haver de recarregar la pàgina ni sortir del llistat.

7.2.1.3Disseny de la funcionalitat

Per detallar com s'ha realitzat la funcionalitat s'indicarà una il·lustració amb la configuració de la vista realitzada i a continuació es detallarà la configuració de cada bloc.



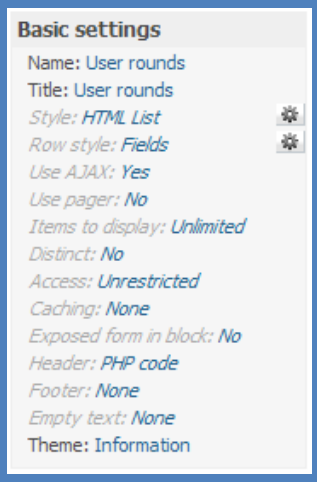
Il·lustració 47 – Configuració vista de les partides d'usuari

Un problema que ens hem trobat ha sigut que el mòdul Views permet que cada element del llistat pugui enllaçar a la pàgina del node corresponent però no a que mostrar la informació en la mateixa pantalla. Per resoldre aquest problema ens ajudarem del mòdul Popup[30], que ens permetrà crear popups (dins de la mateixa pàgina sense obrir cap finestra o pestanya del navegador) que s'obriran al clicar damunt de cada element del llistat mostrant la puntuació de la partida de golf.

A continuació mostrarem una taula amb els detalls de configuració de cada bloc de la vista.

[30] Més informació del mòdul Popup a: http://drupal.org/project/popup

Bloc de configuració	Configuració
Fields (Campo de golf) Node: Title Customfield: PHP code	Els camps que s'han afegir a la vista són: - El nom del camp de golf de la partida - Camp de tipus "customfield" que mitjançant PHP afegirem el popup amb la puntuació de la partida: `<?php print popup_filter_process_text( '[popup node='.$data->nid.' activate=click]' ); ?>` Cridarem a la funció popup_filter_process_text del mòdul Popup indicant que el contingut del popup serà el node que correspongui a la partida de golf i que s'activarà amb un click a sobre del títol del node.
Sort criteria Node: Updated date desc	El resultat de la vista estarà ordenat per la data d'edició del node de forma descendent.
Filters Node: Type = Partida de golf Content: Campo de exposed	S'ha definit un filtre perquè els nodes siguin únicament de tipus "Partida de golf". També s'afegirà el filtre del camp de golf que permetrà als usuaris que apareguin partides de determinats camps de golf. Aquest filtre s'haurà d'exposar a l'usuari.
Relationships Content: Golf course Content: Players	Es definiran dues relacions. La primera serà amb el camp de golf per poder accedir al títol del camp. La segona relació serà per obtenir els jugadors de la partida
Arguments (Players) User: Uid	En aquest bloc definirem el paràmetre amb l'identificador de l'usuari de la sessió utilitzant la relació amb els jugadors de la partida definida a dalt per a només mostrar les partides que ha jugat l'usuari.



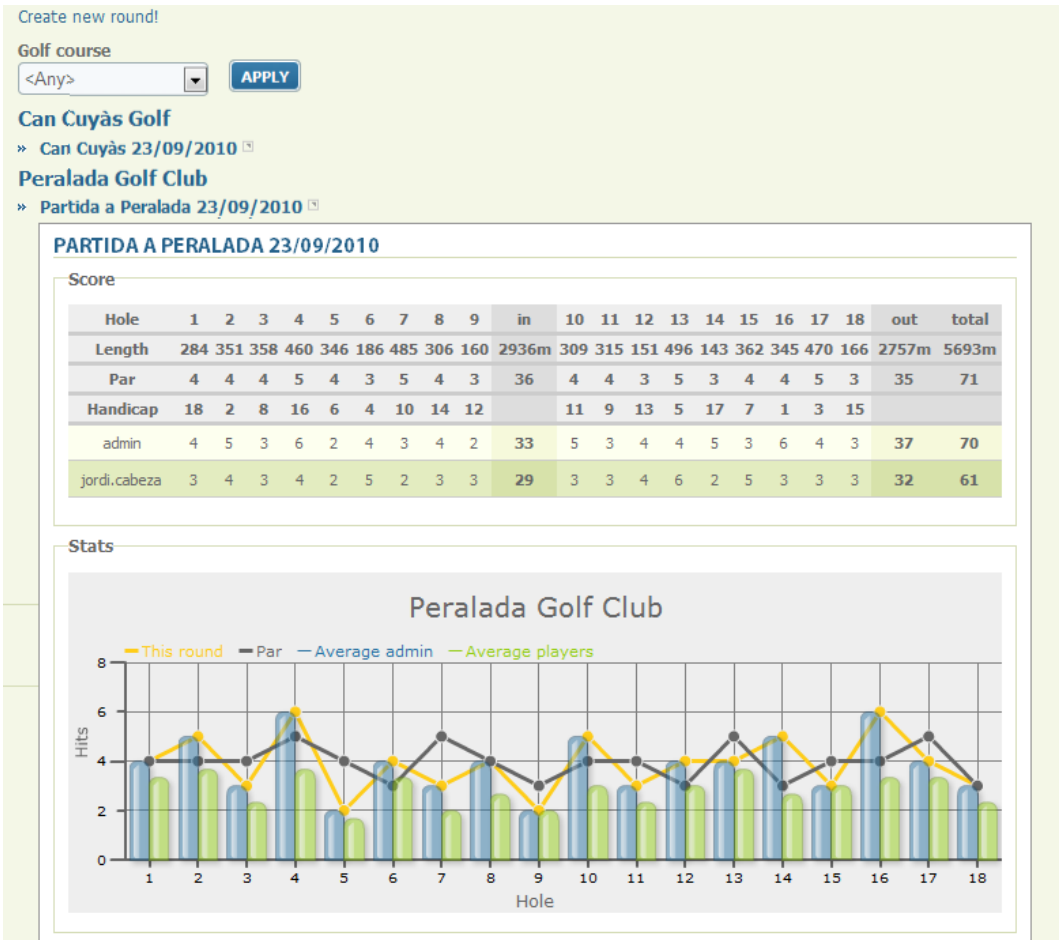
Finalment, en el bloc "Basic settings" s'ha indicat que l'estil de la vista sigui en format de llista HTML.

Per poder crear noves partides s'ha decidit afegir un enllaç a dins la capçalera de la vista en format PHP.

```php
<?php
print(
    l('Create new round!', 'node/add/golf-round')
    );
?>
```

Tornarem a utilitzar la funció "l" per a crear l'enllaç per afegir noves partides.

Un cop configurada la vista el resultat que obtenim és el següent:

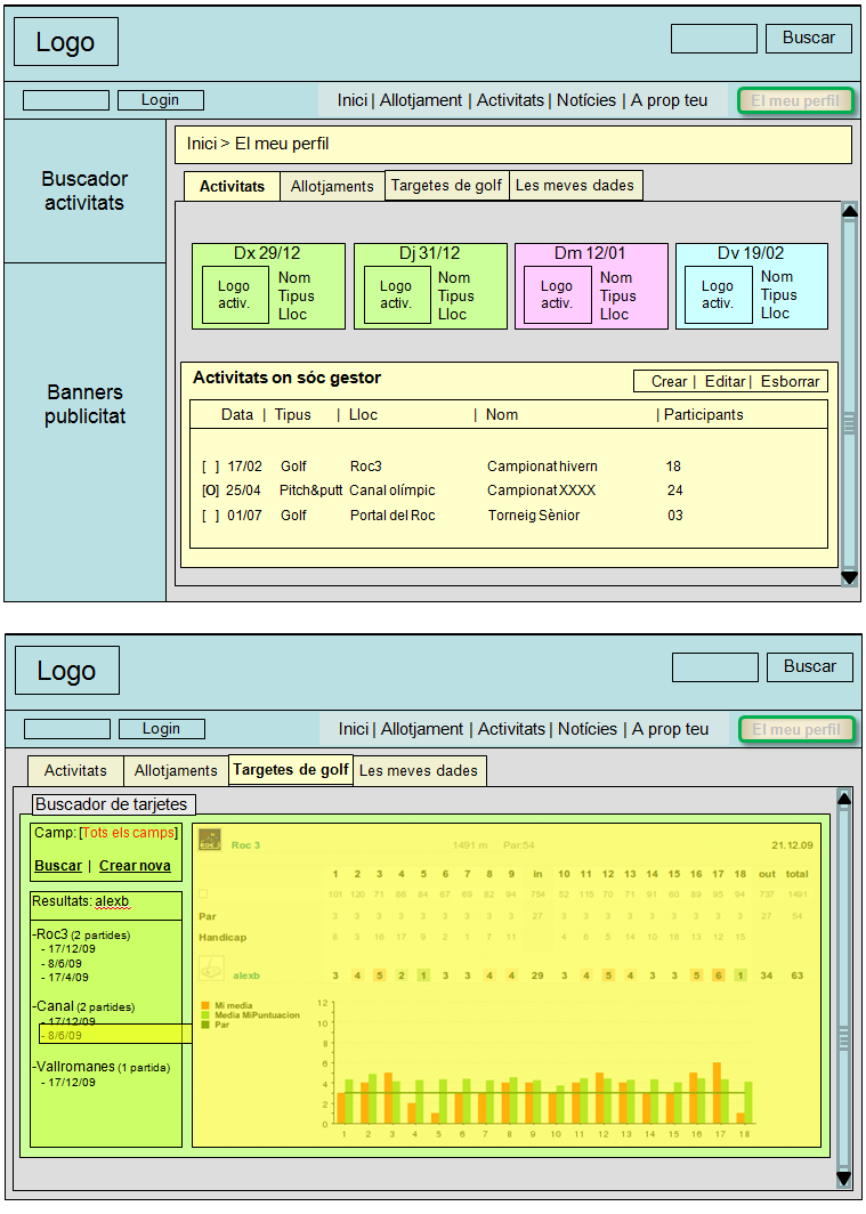


Il·lustració 48 – Vista de les partides de golf de l'usuari

7.2.2 Creació del perfil d'usuari

Aquesta funcionalitat pretén agrupar tota la informació relacionada de l'usuari en una única pàgina. Aquesta pàgina haurà de contenir la informació personal de l'usuari, informació de les cases rurals que ha llogat, informació sobre les activitats que ha participat o creat així com poder veure el resultat de les partides de golf. En aquest projecte oferirem la informació de les activitats i dels resultats de les partides de golf.

7.2.2.1Maqueta de la funcionalitat



Il·lustració 49 – Maqueta "El meu perfil". Bloc d'activitats i de targetes de golf

7.2.2.2Especificació detallada

Del disseny inclòs a l'apartat anterior i de les reunions amb el client obtenim:

1- La informació de "El meu perfil" es presentarà mitjançant pestanyes. Cada pestanya ens mostrarà la informació d'un dels 4 blocs d'informació:
    a. Activitats
    b. Allotjaments
    c. Targetes de golf
    d. Dades personals de l'usuari
2- El bloc d'Activitats haurà de contenir la vista de les activitats on participo i la vista de les activitats que gestiono.
3- El bloc de targetes de golf contindrà la vista creada en el punt anterior: la vista de les partides de golf de l'usuari
4- S'haurà de poder accedir des de qualsevol punt del sistema al perfil de l'usuari. Per tant, s'inclourà un enllaç visible des de qualsevol pàgina.

7.2.2.3Disseny de la funcionalitat

Per crear pestanyes en Drupal disposem de diversos mòduls que ens poden ajudar. Després de realitzar un estudi de les diferents opcions que es poden trobar s'ha decidit utilitzar el mòdul Quick Tabs[31]. Aquest mòdul ens permet crear blocs de contingut separats per pestanyes, justament la forma amb la que volem presentar el contingut. S'ha escollit aquest mòdul degut a la seva facilitat d'ús, per permetre adaptar l'estil de les pestanyes i per disposar de força documentació i problemes resolts ja que és un mòdul força utilitzat per la comunitat de desenvolupadors de Drupal.

Per assolir la funcionalitat de crear el perfil d'usuari dividirem la feina en 4 tasques:

1- Configuració del mòdul Quick Tabs i creació de la pàgina on s'ubicarà el perfil
2- Afegir el bloc de continguts d'activitats
3- Afegir el bloc de les targetes de golf

[31] Més informació del mòdul Quick Tabs a: http://drupal.org/project/quicktabs

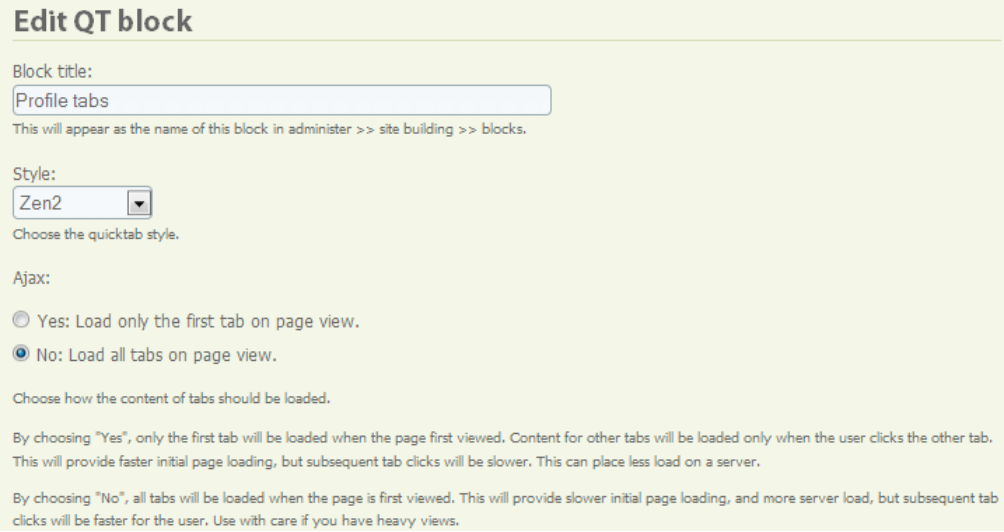
4- Creació d'un botó/enllaç per accedir al perfil d'usuari des de qualsevol pàgina del portal

Configuració del mòdul Quick Tabs i creació de la pàgina on s'ubicarà el perfil

La primera tasca serà la creació d'una pàgina on allotjarem les pestanyes. Recordem que una pàgina és un dels tipus de continguts predefinits per Drupal, i per tant, un node. Per a aquesta pàgina únicament definirem el títol ("El meu perfil") i indicarem quin path volem que tingui la pàgina per no haver d'accedir com si fos un node normal ("domini_portal/node/id_node"). En el nostre cas indicarem que el path sigui "my_profile" , així al indicar a Drupal la url "domini_portal/my_profile" accedirem a la pàgina que acabem de crear.

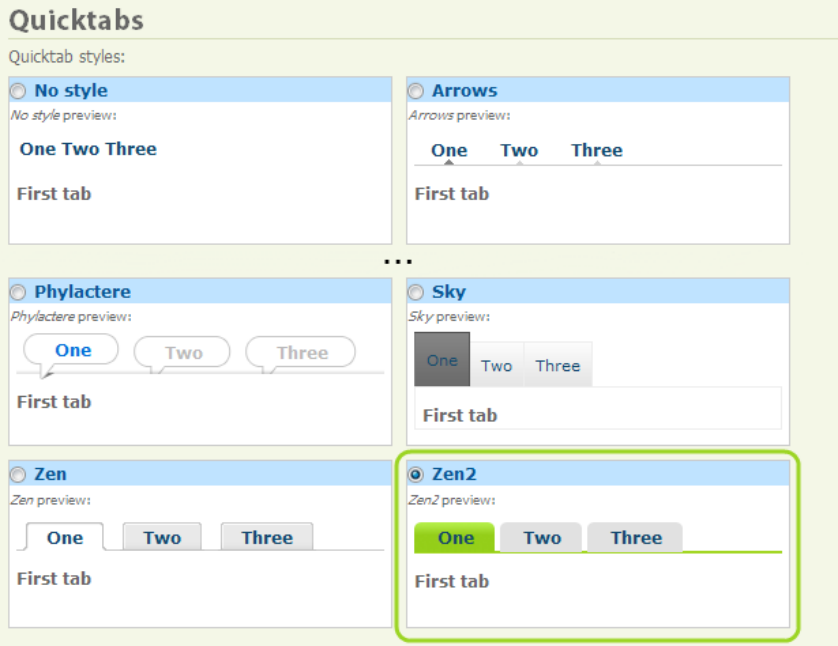
Il·lustració 50 – Creació de la pàgina "El meu perfil"

Un cop creada la pàgina, crearem el bloc de pestanyes amb Quick Tabs. Les pestanyes es crearan en un bloc de Drupal que posteriorment podrem afegir a la pàgina que acabem de crear mitjançant la pàgina de configuració dels blocs. Al crear el bloc de pestanyes indicarem el seu títol, l'estil HTML+CSS de les pestanyes, si volem que s'utilitzi Ajax (s'anirà obtenint el contingut de cada pestanya quan l'usuari vulgui consultar la informació) o carregar tot el contingut.



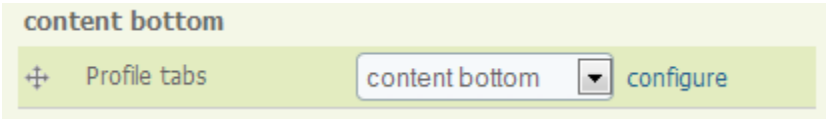
Il·lustració 51 – Configuració del bloc amb les pestanyes del perfil d'usuari

Per tal que les pestanyes es visualitzin amb un estil uniforme amb la resta del portal s'ha creat un nou estil a partir d'un dels estils existents que proporciona quicktabs. Per afegir un nou estil s'ha de crear una nova carpeta a "sites/all/modules/quicktabs/tabstyles" (on el mòdul guarda els estils) amb els fitxers de les imatges i els fitxers CSS necessaris (han de tenir el nom de la carpeta). El propi mòdul quicktabs a la pantalla de configuració s'adonarà de l'existència d'un nou estil i el llistarà com un dels possibles a seleccionar tal com mostrem en la següent imatge. L'estil creat és el que es mostra dins del rectangle verd. S'ha intentat que tingui un aspecte que s'integri amb l'estil del portal sense semblar que sigui un afegit creat per un mòdul extern.

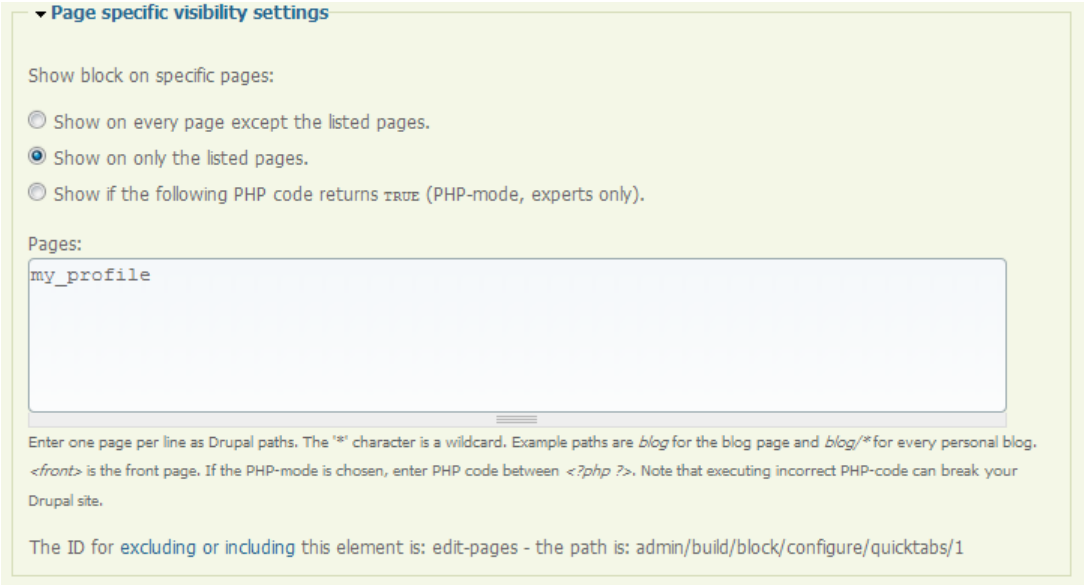


Il·lustració 52 – Creació d'un estil personalitzat per mostrar les pestanyes

Un cop tenim la pàgina creada i el bloc amb les pestanyes creat i configurat ja podem afegir el bloc de pestanyes a la pàgina. Per això ens dirigirem a la pàgina d'administració de blocs ("admin/build/block/list") i indicarem que volem mostrar el bloc en la zona "Content Bottom" (zona central sota el títol de la pàgina).



En la configuració indicarem que només el volem mostrar a la pàgina de "El meu perfil". D'aquesta manera sempre que anem a la pàgina del perfil d'usuari veurem el bloc amb les pestanyes.



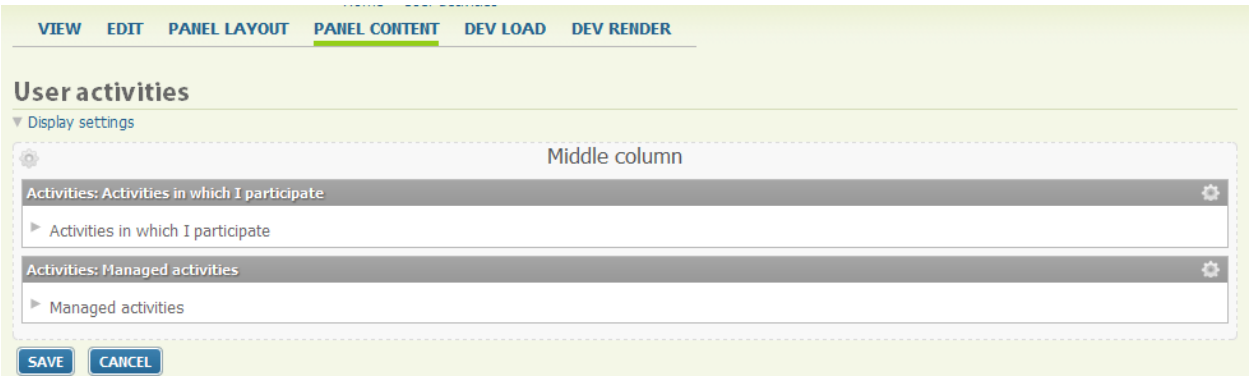
Ara ja tenim tot gairebé preparat per a poder visualitzar el perfil de l'usuari. Únicament ens falta afegir el contingut de cada pestanya.

Afegir el bloc de continguts d'activitats

Per afegir el bloc de continguts d'activitats, que ha de contenir la vista de les activitats on participo i la vista de les activitats que gestiono, primer haurem de crear un contingut que permeti allotjar aquestes dues vistes en un única pàgina. Per fer-ho ens ajudarem del mòdul Panels[32] que ens permetrà disposar d'un nou tipus de contingut en el sistema, el tipus de contingut Panel. La idea és que un "panel" serveix de recipient on poden afegir contingut amb diverses opcions de maquetació (diverses columnes, una columna, ...).

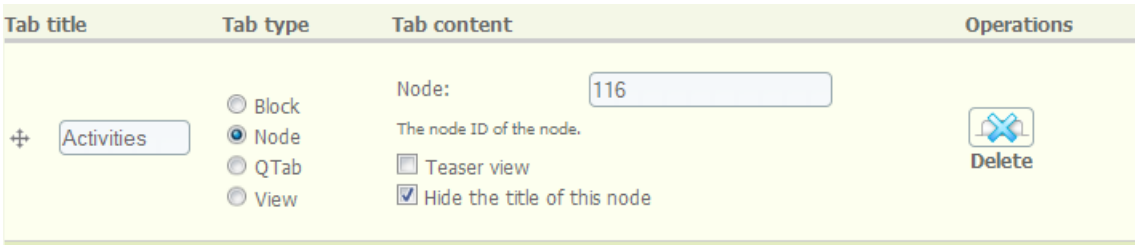
Llavors crearem un node de tipus "panel" d'una única columna on afegirem les dues vistes creades en iteracions anteriors relacionades amb les activitats del usuari. En la següent imatge es pot veure com queda la configuració del "panel", amb les dues vistes afegides a l'única columna disponible.

[32] Més informació del mòdul Panels a: http://drupal.org/project/panels



Il·lustració 53 – Pantalla de configuració del contingut d'un "panel

Un cop tenim el node de tipus panel creat ens dirigirem un altre cop a la pantalla d'edició del bloc de pestanyes i afegirem el panel com una nova pestanya. Per fer-ho únicament haurem d'indicar que la pestanya serà de tipus node i indicar l'identificador numèric del node on hem creat el "panel".



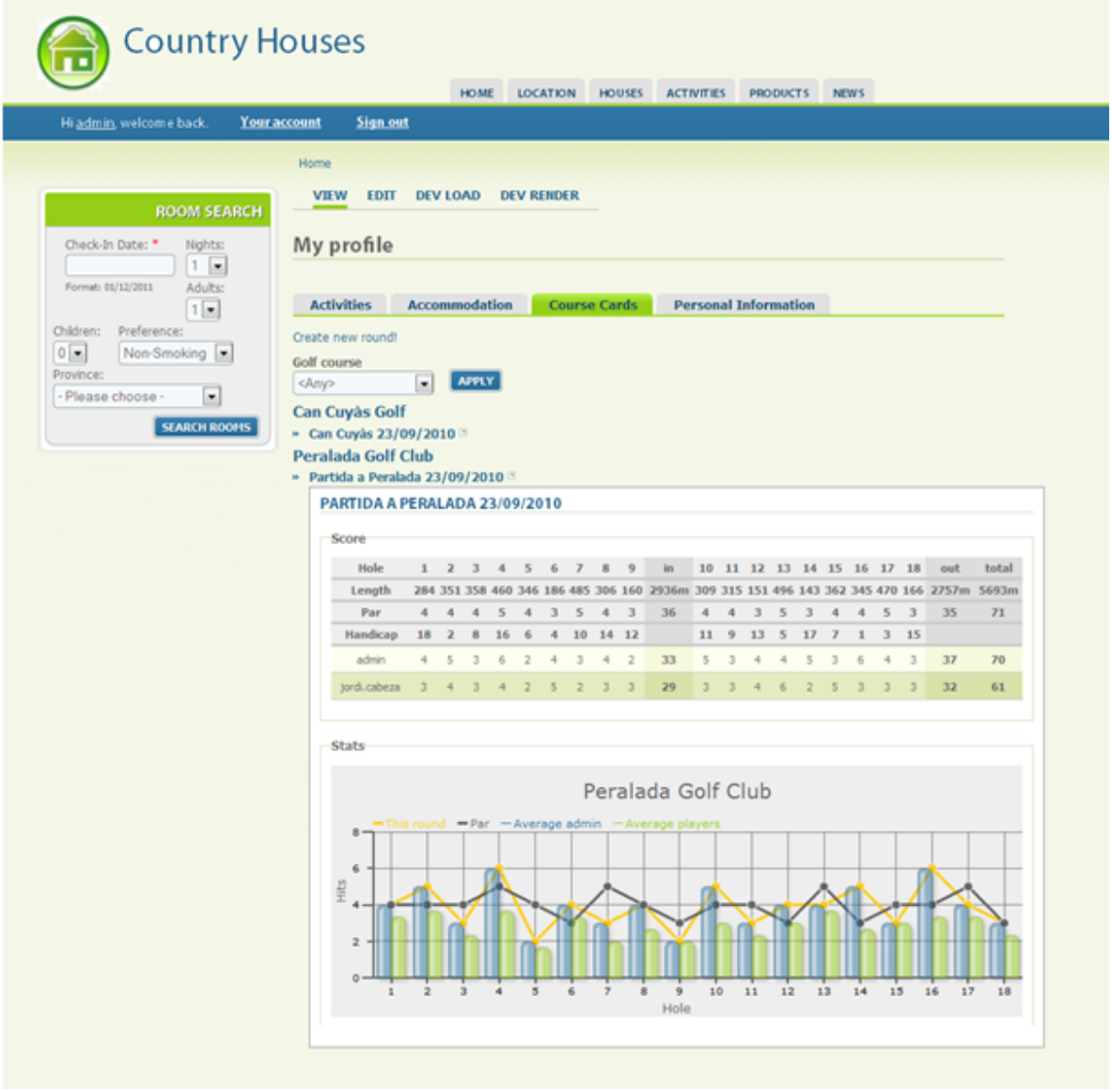
Un cop afegida la pestanya, ens podem dirigir a la pàgina de "El meu perfil" ("domini_drupal/my_progfile") on podrem visualitzar que ja tenim una pestanya afegida a la pàgina amb el contingut de les activitats de l'usuari.

Il·lustració 54 – Pestanya d'activitats de "El meu perfil"

Afegir el bloc de les targetes de golf

Per afegir el bloc targetes únicament haurem d'anar a la pàgina de configuració del bloc de pestanyes que s'ha creat amb QuickTabs i afegir una nova pestanya. S'haurà d'indicar que el contingut es de tipus "View" i indicar la vista que volem mostrar.

Amb la pestanya creada ja es poden veure les partides de l'usuari en la secció "El meu perfil".



Il·lustració 55 – Pestanya de targetes de golf de "El meu perfil"

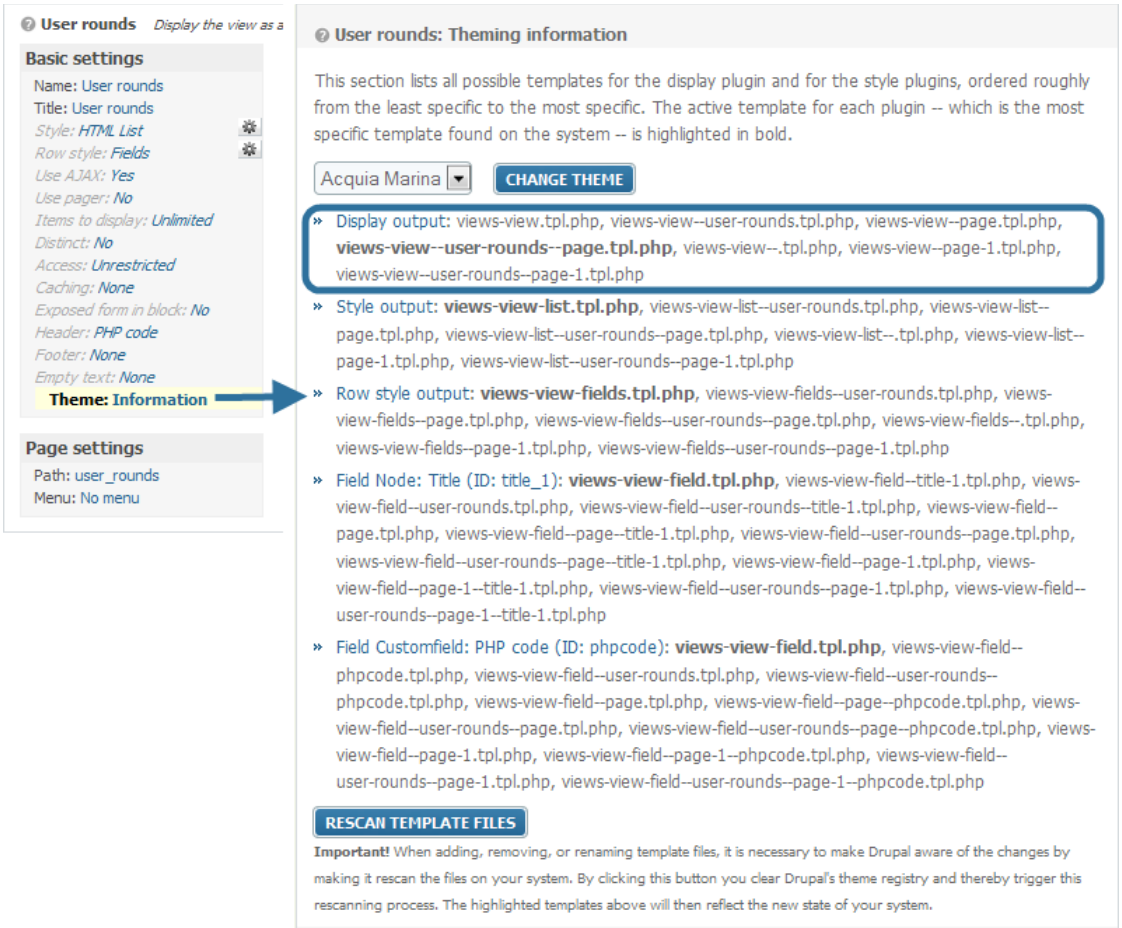
Fent proves del funcionament de la vista dins la pestanya es va trobar que quan es filtrava el resultat per un camp de golf i s'actualitzava el llistat de partides no funcionaven els popups amb la puntuació de la partida.

Després d'investigar es va trobar que el problema era el codi jQuery que permetia al clicar obrir el popup no actualitzava els popups quan es recarregava via AJAX el resultat de la llista. Per solucionar aquest problema s'ha creat un template específic per a aquesta vista idèntic al genèric afegint el codi jQuery que permet que es puguin obrir els popups.

```
<script> $('.popup-element').popup();</script>
```

Perquè Drupal executi aquesta nova plantilla enlloc de la genèrica per a aquesta vista l'haurem d'ubicar a la carpeta "sites/all/themes/nom_del_tema/views/" amb la següent nomenclatura "views-view--nom-vista--tipus-visualitzacio.tpl.php". En el nostre cas s'anomenarà "views-view--user-rounds--page.tpl.php".

Amb el fitxer ubicat ens dirigirem a la pantalla de configuració de la vista i accedirem a la secció d'informació del tema. Un cop allà seleccionarem l'opció "Rescan template files" perquè Drupal s'assabenti si s'ha creat un nova plantilla que sobreescriu la genèrica per a aquesta vista. En aquest cas, hauria de trobar la plantilla que acabem de crear tal com es mostra a continuació.



Il·lustració 56 – Informació de les plantilles de la vista de partides de golf

Creació botó "El meu perfil"

Amb la secció de "El meu perfil" creada només cal afegir un enllaç visible en tot el portal perquè els usuaris puguin accedir al seu perfil.

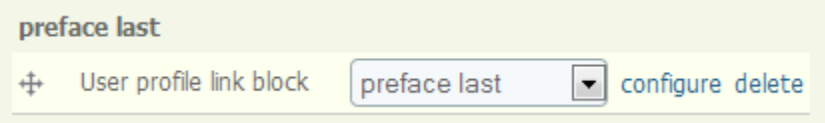
Aquesta només ha d'estar visible un cop l'usuari s'hagi autenticat en el sistema, d'altra manera, seriem incapaços de saber de quin usuari.

Per fer-ho crearem un bloc que ens mostri un botó que al clicar-hi a sobre ens portarà al la pàgina del perfil d'usuari.

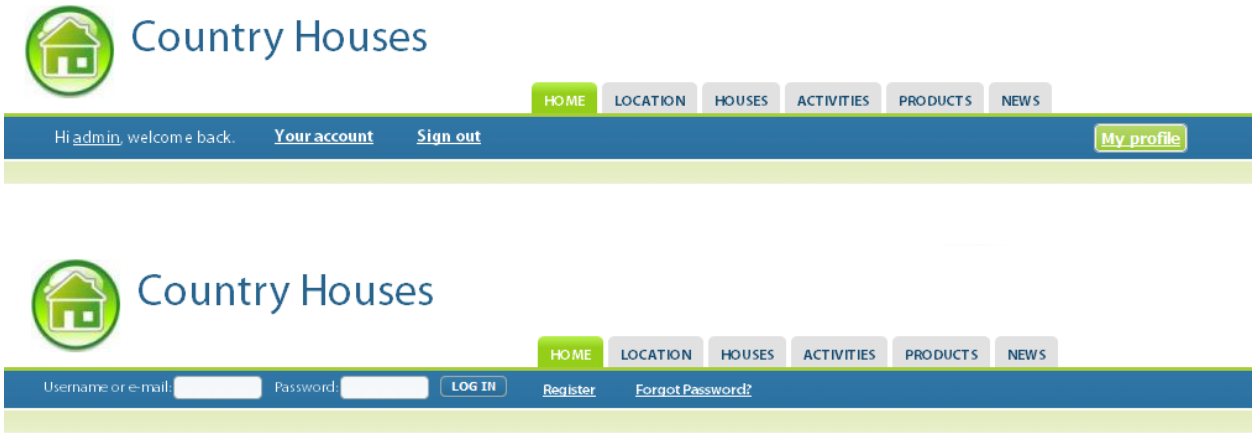
El contingut del bloc serà el següent codi PHP que generarà el codi HTML per veure el botó de la següent imatge que serà un enllaç a "El meu perfil". En l'apartat de configuració s'indicarà que sigui visible a tot el portal només per usuaris autenticats. A part, s'ha hagut de definir l'estil CSS del botó.

Il·lustració 57 – Configuració del bloc amb enllaç a "El meu perfil"

Ara únicament cal ubicar el bloc amb el botó a la secció de la pantalla on el vulguem mostrar. En el nostre cas l'ubicarem en la secció l'ubicarem "preface_last" que correspon a la dreta de la secció on es mostra el login d'usuari.



A continuació veurem com queda la capçalera del portal amb el bloc amb l'enllaç a "El meu perfil". En la imatge superior trobem el cas que l'usuari està autenticat mentre que en la inferior al no estar autenticat no es mostra el botó creat.



Il·lustració 58 – Capçalera portal amb el botó per accedir al perfil d'usuari

8 Iteració 5

8.1 Llista d'activitats de la iteració

8.1.1 Unificació projectes paral·lels Drupal

1- Realitzar una instal·lació de Drupal partint des de cero.
2- Definir els fitxers modificats en el desenvolupament.
3- Definir un llistat de mòduls utilitzats pel portal Drupal.
4- Definir les configuracions dels mòduls utilitzats.
5- Definir procediment per fer la migració del portal.
6- Executar la migració per obtenir un únic portal a partir dels 2 projectes paral·lels desenvolupats sobre el portal.

8.1.2 Creació de serveis per aplicacions externes

1- Estudiar com definir mètodes accessibles per un client extern (dispositiu Android) per accedir a la informació del Drupal.
2- Definir serveis necessaris perquè l'aplicació mòbil pugui gestionar les partides de golf dels usuaris.
3- Creació dels serveis definits en el punt anterior.
4- Provar els serveis definits assegurant que es port realitzar la comunicació amb el portal Drupal.

8.2 Anàlisi i disseny de les funcionalitats

8.2.1 Unificació projectes paral·lels Drupal

En aquest punt es pretén integrar el contingut realitzat pels dos projectes que s'han desenvolupat sobre la mateixa plataforma.

En aquest document s'explicarà com instal·lar un portal Drupal des de cero per després afegir el que s'ha realitzat en aquest desenvolupament.

En les diferents reunions desenvolupades s'ha decidit que l'ordre de la migració serà afegir primer els desenvolupaments i configuracions d'aquest projecte i després afegir el desenvolupament. Aquest ordre és degut a que aquest desenvolupat involucra més mòduls i configuracions resultant més ràpid partir de la base de dades d'aquest desenvolupament (on a part del contingut són guardades les configuracions dels mòduls).

A continuació detallarem el procés que s'ha seguit per realitzar la migració.

8.2.1.1 Preparació de l'entorn

Com s'ha definit en el punt 1.3.1 d'aquest document, on es defineix la torre de funcionament del Drupal, necessitem el disposar dels següents per fer funcionar el sistema:

- Sistema Operatiu (en aquest cas s'utilitzarà Windows 7)
- Intèrpret de PHP
- Gestor de bases de dades
- Servidor web

Per als últims tres punts del llistat s'instal·larà el paquet de software WAMP[33] que ens proporciona la infraestructura necessària incloent intèrpret de PHP, gestor de bases de dades MySQL i servidor web Apache.

La versió de WAMP instal·lada incorpora PHP 5.2.9-2, MySQL 5.1.33 i Apache 2.2.11.

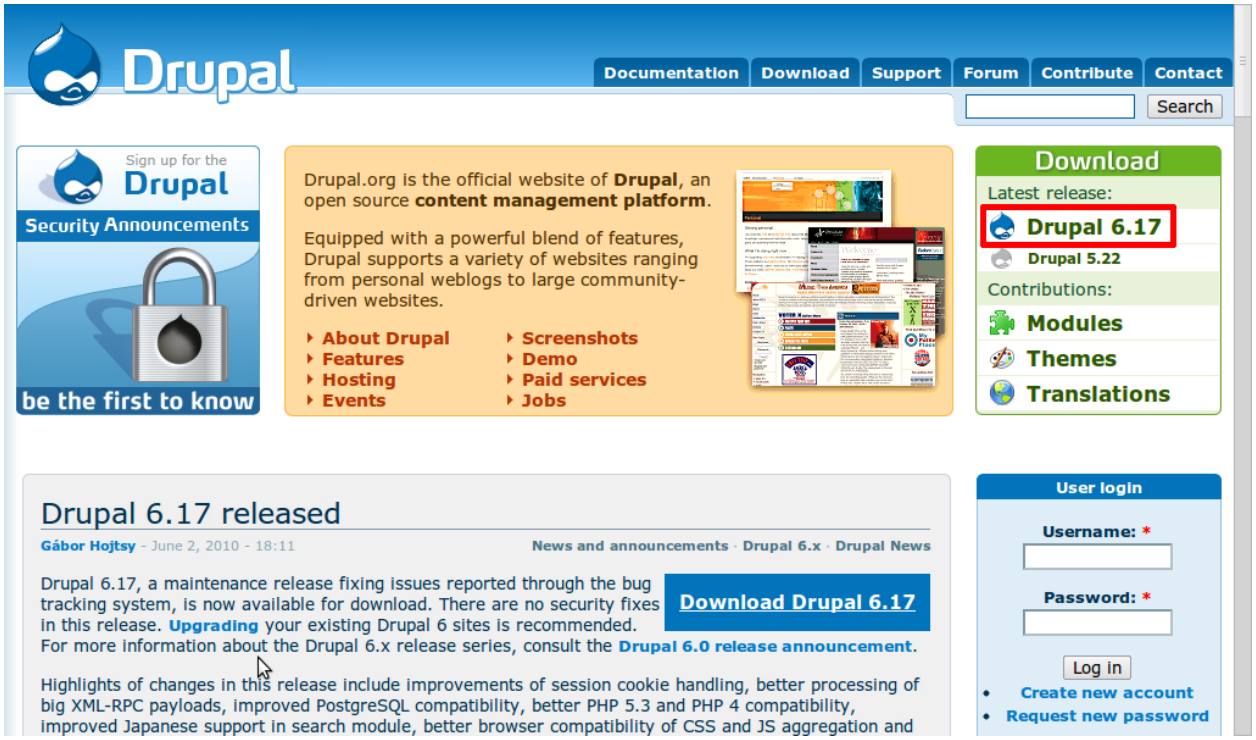
Amb l'entorn preparat ja podem instal·lar Drupal.

[33] Més informació sobre WAMP a: http://es.wikipedia.org/wiki/WAMP

8.2.1.2 Instal·lació de Drupal i mòduls necessaris

En aquest apartat s'explicarà com instal·lar el desenvolupament realitzat sobre Drupal en aquest projecte en una nova màquina. Aquest desenvolupament està format pels mòduls (fitxers de codi i configuració) utilitzats, el tema utilitzat, la base de dades que anirà adjunta al projecte i els fitxers dels continguts (com poden ser lest imatges).

Primer de tot, s'ha de descarregar una versió de la plataforma de gestió de continguts disponible a la pàgina web oficial de Drupal (http://www.drupal.org). En el moment de realitzar aquesta instal·lació l'última versió oficial és la 6.17. El projecte desenvolupat és compatible amb qualsevol versió 6 de Drupal.

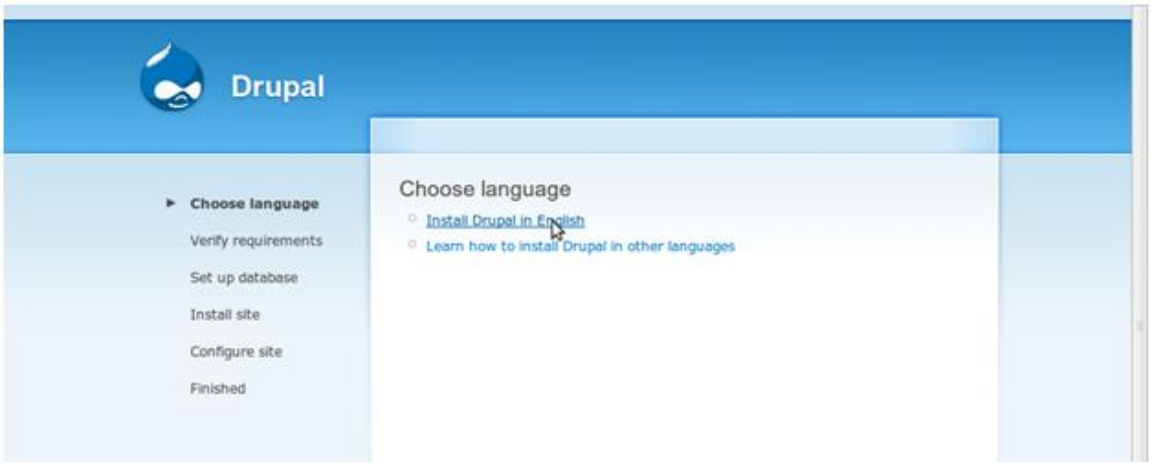


Il·lustració 59 - Pàgina principal de drupal.org

Un cop descarregat el codi font del gestor de continguts, aquest s'hà de descomprimir en un directori accessible pel servidor web. Dintre del directori on s'instal·la WAMP es troba el directori "www" on podem allotjar el sistema de fitxers de Drupal. Per accedir-hi haurem

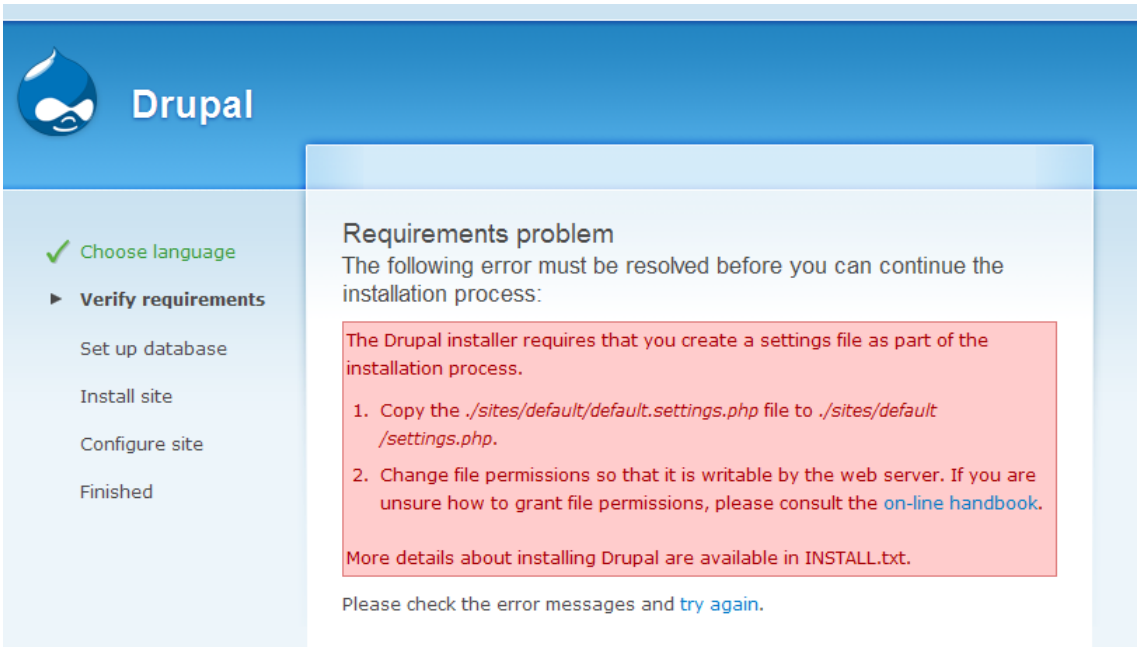
d'introduir el nom de la màquina al navegador seguit del nom de la carpeta del Drupal. Per exemple, http://localhost:8080/drupalPFC si estem en local.

Accedint a aquesta direcció ens trobarem amb el primer pas de la instal·lació. En aquest pas se'ns demarà decidir en quin idioma volem utilitzar per defecte Drupal. Es poden afegir idiomes descarregant-los de la pàgina oficial. Aquesta instal·lació la farem en anglès, donat que és l'idioma que es va utilitzar en la instal·lació del projecte.



Il·lustració 60 - Primer pas d'instal·lació de Drupal

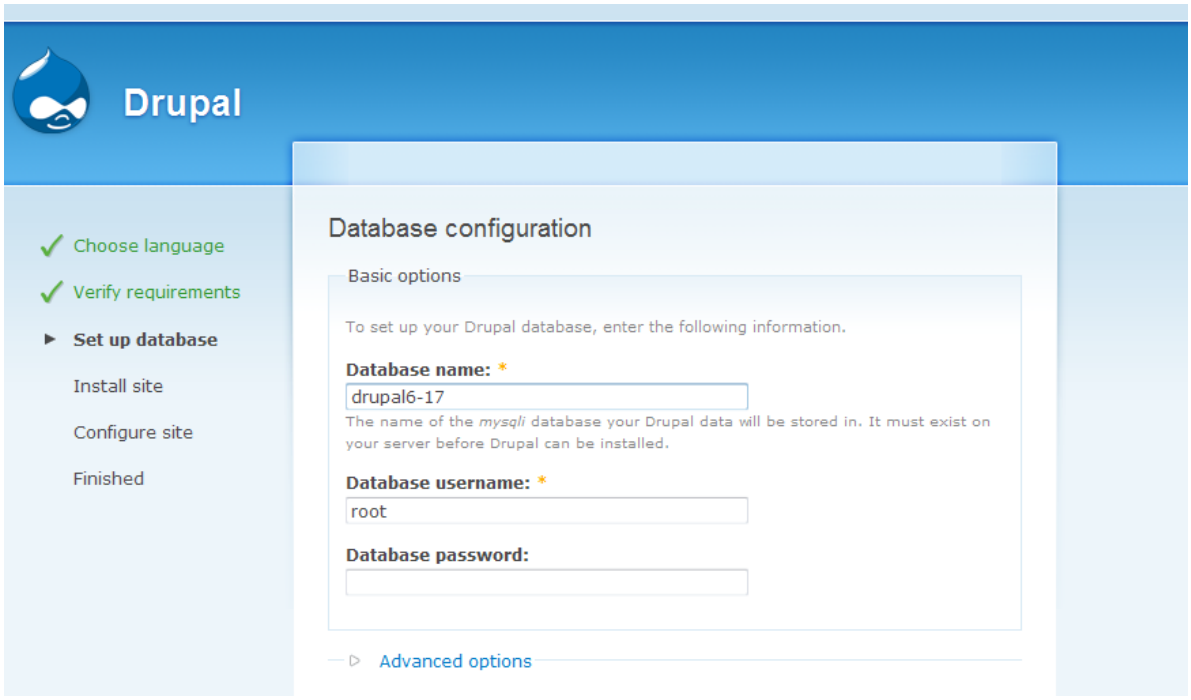
Al clicar l'opció "Install Drupal in English" el programa d'instal·lació comprovarà si es pot instal·lar el sistema. En el nostre cas (ja que no hem tocat cap fitxer del sistema), sortirà el següent missatge:



Per poder solucionar aquest problema s'han de seguir les següents indicacions:

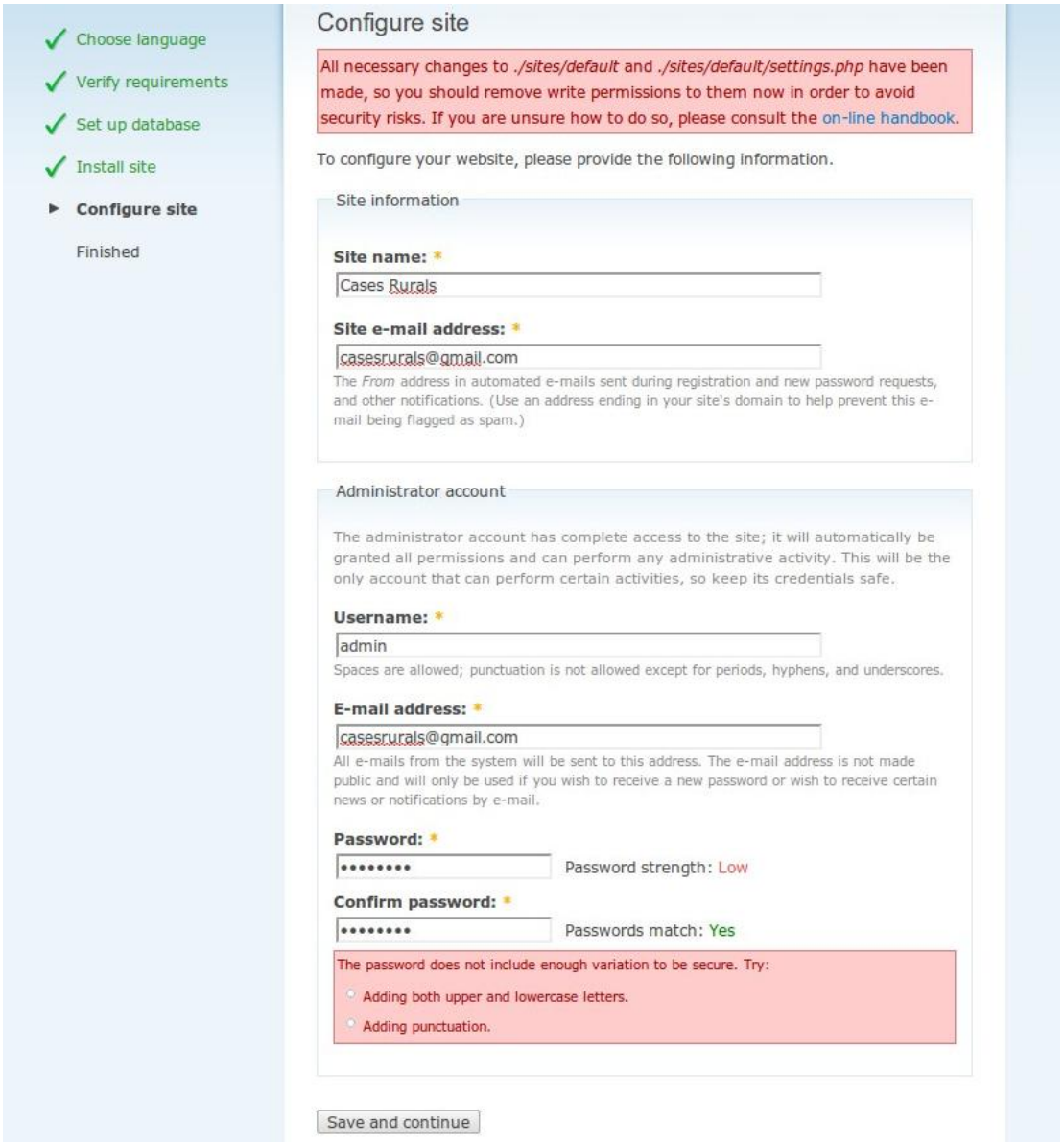
1- Crear el fitxer <Directori_Drupal>/sites/default/settings.php copiant-lo de <Directori_Drupal>/sites/default/default.settings.php i
2- Donar-li permisos d'escriptura al servidor web al sistema de ftixers del Drupal.

Després d'aquests dos passos podrem anar al següent punt de la instal·lació: la configuració de la base de dades.



El sistema ens demana el nom de la base de dades i un usuari per entrar i editar aquesta base de dades. La base de dades introduïda ha d'estar ja creada abans de realitzar aquest pas. Ho podem fer desde la consola de MySQL amb la comanda "CREATE DATABASE nom_base_dades;".

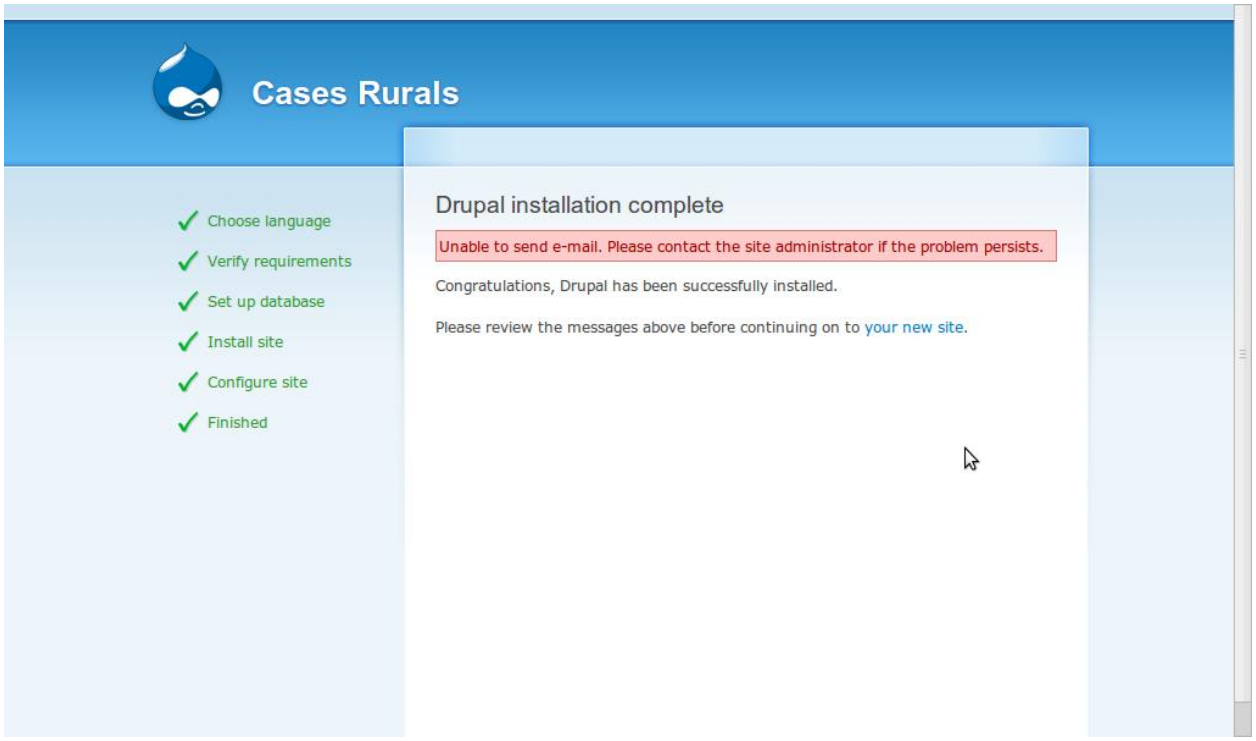
Amb la base de dades creada correctament passarem a l'últim pas de la instal·lació.



En aquest últim pas s'han d'escriure les dades bàsiques de la portal web:

- Nom del portal
- Direcció de correu del portal
- Usuari administrador

Quan s'hagi omplert el formulari i premut el botó de guardar el sistema ens mostrarà una pantalla avisant que la instal·lació ha finalitzat correctament.



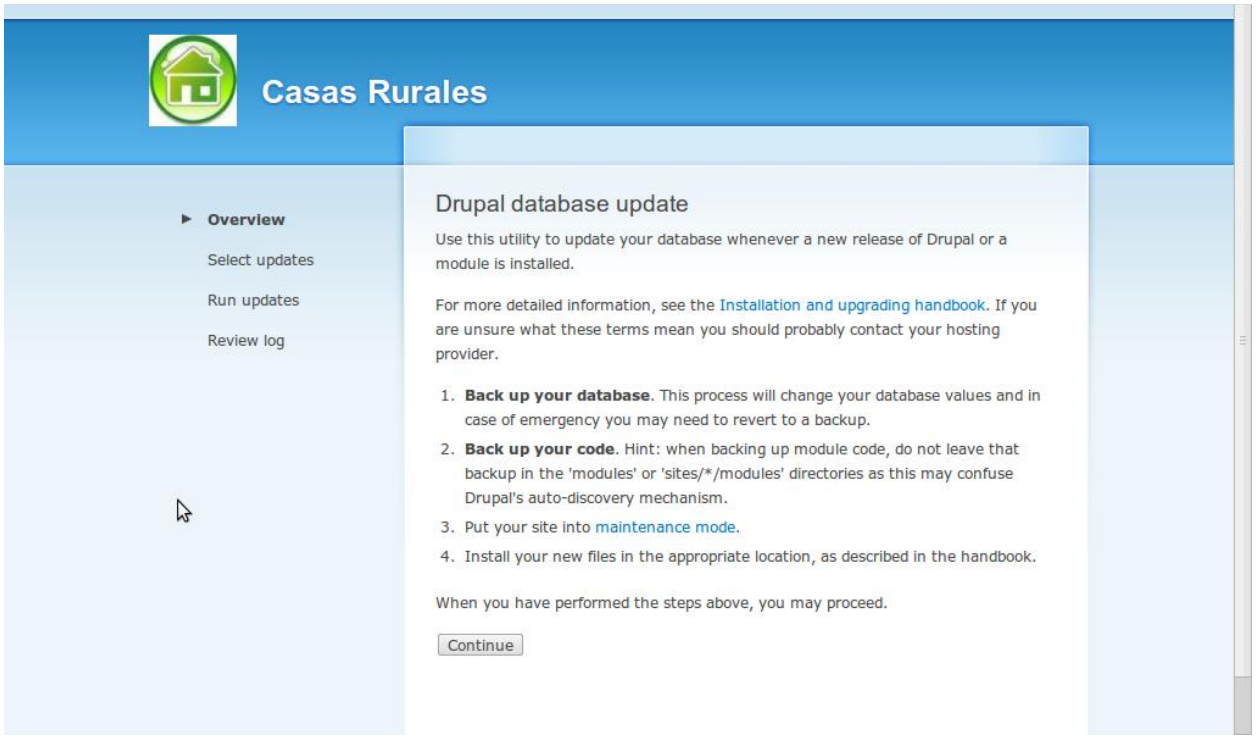
Il·lustració 61 - Últim pas d'instal·lació de Drupal

Un cop finalitzada la instal·lació del Drupal, procedirem a copiar els fitxers necessaris del nostre portal. Concretamment haurem de copiar el tema, els mòduls utilitzat i els fitxer pujats al portal pels usuaris:

- ***Theme***: Afegirem el tema que s'ha utilitzat (*acquia_marina*). S'haurà d'activar més tard a la pantalla d'administració de temes. Copiar a <Directori_Drupal>/sites/all/themes.
- ***Modules***: Afegirem els mòduls no core que necessita el portal per funcionar correctament al directori <Directori_Drupal>/site/all/modules
- ***Files***: Afegir totes les imatges dels continguts creats pels usuaris. S'han de copiar al director <Directori_Drupal>/sites/default/files. El servidor ha de tenir els permisos suficients per poder crear i modificar fitxers.
- drupal_pfc.sql: Fitxer que conté l'últim backup de la base de dades incloent els usuaris, contingut i configuracions (mòduls, vistes, tipus de contingut,...). Per a poder importar aquest fitxer executarem la següent comanda des de la consola del mysql:

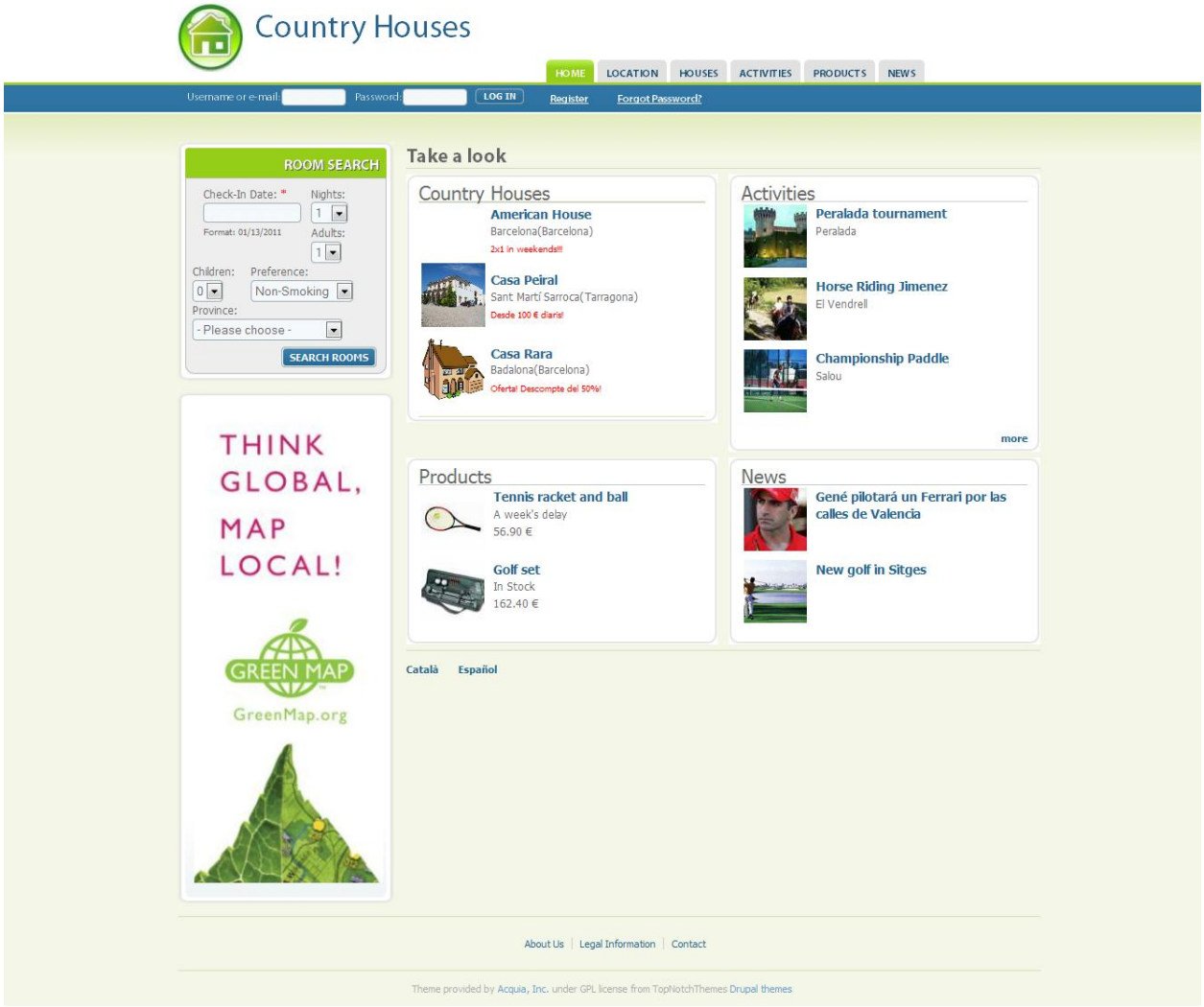
$ nom_base_dades_destí < drupal_pfc.sql;

Després de fer tot el proces d'instal·lació si anem a http://localhost:80/drupal6-17/update.php i ens trobarem la següent pantalla:



Il·lustració 62 - Primer pas de l'actualització de programari del Drupal

Aquesta pantalla és el primer pas per actualitzar els mòduls i el programari bàsic de la instal·lació actual del Drupal. Al finalitzar l'actualització estarem llestos per anar a http://localhost:80/drupal6-17/ i veure la pàgina inicial del portal.



Il·lustració 63 – Pàgina principal del portal

Per acabar la instal·lació únicament fa falta realitzar una petita modifficació. Respecte a la instal·lació original hi va haver un petit canvi en la gestió d'usuari en la primera versió del sistema, així que si s'intenta entrar a la gestió d'usuaris es provocarà un error general en el portal Drupal. El resultat serà una pantalla en blanc (amb errors en el fitxer bootstrap.inc en cas de tenir el errors activat tant en el servidor PHP com en el Drupal).

Per solucionar aquest error, es pot copiar el fitxer bootstrap.inc que hi ha dintre de Files.tar.gz a la carpeta <Directori_Drupal>/include o anar al fitxer <Directori_Drupal>/include/bootstrap.inc i modificar la línea 794 per el següent codi PHP:

```
function drupal_unpack($obj, $field = 'data') {
  if ($obj->$field && $data = unserialize($obj->$field)) {
    foreach ($data as $key => $value) {
      if (!isset($obj->$key)) {
        $obj->$key = $value;
      }
    }
  }
  return $obj;
}
```

bootstrap.inc abans de la modificació

```
function drupal_unpack($obj, $field = 'data') {
  if ($obj->$field && $data = unserialize($obj->$field)) {
    foreach ($data as $key => $value) {
      if ($key && !isset($obj->$key)) {
        $obj->$key = $value;
      }
    }
  }
  return $obj;
}
```

bootstrap.inc modificat

Després de modificar aquest fitxer només queda comprovar que es troben actius tots els mòduls necessaris pel correcte funcionament del portal. Per fer-ho s'ha d'anar a la pantalla d'administració de mòduls http://localhost:80/admin/build/modules/list.

A continuació indiquem un llistat dels mòduls utilitzats. En el primer nivell trobem el nom del grup de mòduls, en el següent el nom del mòdu, i en el tercer els submòduls (si en té el mòdul) que es troben actius.

1. Administration
    1. Administration menu
2. Other
    1. Format Number API
    2. Addresses
    3. Addresses CCK
    4. Addresses - Users
    5. Token
    6. Advanced help
    7. Area Banner

8. CSS Injector
9. FancyZoom
10. FCKEditor
11. IMCE
12. Lightbox2
13. LoginToboggan
14. Menu breadcrumb
15. Quick Tabs

3. CCK
    1. Conditional fields
    2. Content
    3. Content Copy
    4. Fieldgroup
    5. Node Reference
    6. Number
    7. Option Widgets
    8. Text
    9. User Reference
    10. Content Taxonomy
        1. Content Taxonomy
        2. Content Taxonomy Autocomplete
        3. Content Taxonomy Options
    10. Content Templates
    11. FileField
    12. Formatted Number CCK
    13. Google Maps Embed
    14. ImageField
    15. Link
    16. Money CCK field
    17. Node Relationships
    18. Videofield
4. Currency
    1. Currency API

   Views
    1. Views
    2. Views exporter
    3. Views UI
    4. Bonus: Paged Feed
    5. Bonus: Panels
    6. Bonus: Views Export
    7. Bonus: Views Spy
    8. Views Custom Field
    9. Views Fluid Grid
6. User Interface

    1. jQuery UI
    2. jQuery Update
7. Modal Frame
    1. Modal Frame API
8. Swftools
    1. Flowplayer
    2. SWF Tools
9. Ctools
    1. Chaos tools
    2. Page manager
10. Date
    1. Date
    2. Date API
    3. Date Popup
    4. Date Timezone
11. Devel (jo el tinc desactivat)
12. ImageCache/ImageAPI
    1. ImageAPI
    2. ImageAPI GD2
    3. ImageCache
    4. ImageCache UI
13. Multilanguage (I18N)
    1. Content type translation
    2. Internationalization
    3. Language Icons
    4. Localization client
    5. String translation
    6. Taxonomy translation
14. Panels
    1. Mini panels
    2. Panel nodes
    3. Panels
15. Google Analytics
16. Taxonomy menu
    1. Taxonomy Menu
    2. Taxonomy Menu Custom Path
    3. Taxonomy Menu Hierarchy
    4. Taxonomy Menu Vocabulary Path
17. Ubercart core
    1. Cart
    2. Conditional Actions
    3. Order
    4. Product
    5. Store

18. Ubercart core - optional
    1. Payment
19. Ubercart extra
20. Ubercart fulfillment
21. Ubercart hotel
    1. Hotel Booking
    2. Hotel Booking - Upgrades & Addons
    3. Hotel Booking Availability
    4. Hotel Room Type
22. Ubercart payment
    1. Credit Card
    2. Skipjack
23. Ubercart uc-sermepa
    1. La caixa
24. User Login Bar
25. Popup
    1. Popup
    2. Popup block
    3. Popup Filter
    4. Popup menu
    5. Popup UI
26. Voting
    1. Voting API
    2. Fivestar
    3. Fivestar Comments

Un cop comprovat que tots els mòduls es troben actius, la plataforma estarà totalment operativa.

Un cop feta la migració d'aquest desenvolupament cal incorporar el codi del desenvolupament paral·lel. Quan els dos projectes estiguin integrats a la mateixa màquina serà aquesta la que utilitzarem per realitzar l'últim desenvolupament al Drupal.

8.2.2 Creació de serveis per aplicacions externes

En aquest punt del projecte, just abans de començar amb el desenvolupament de l'aplicació per a dispositius mòbils, ens ocuparem que des de el mòbil o altres dispositius es pugui accedir a la informació del portal. Es a dir, integrar aplicacions externes amb Drupal. En aquest desenvolupament únicament ens centrarem en la gestió de partides de golf.

8.2.2.1Especificació detallada

Per a poder gestionar les partides de golf mitjançant una aplicació externa, aquesta haurà de ser capaç de poder realitzar les següents operacions al portal Drupal:

1- Autenticar un usuari real.
2- Accedir a la informació dels camps de golf del sistema.
3- Accedir al llistat de possibles jugadors (llistat d'usuaris).
4- Accedir a les partides de l'usuari que s'hagi autenticat en l'aplicació.
5- Permetre actualitzar les puntuacions i jugadors d'una partida.
6- Permetre crear noves partides de golf.

8.2.2.2Disseny de la funcionalitat

Per crear aquesta funcionalitat es podia optar per implementar un sistema independent de Drupal que fes la funció de servidor de serveis i que es comuniqués directament amb la base de dades per donar la informació als client. Per altra banda, es podia buscar solucions existents aportades per la comunitat de desenvolupadors de Drupal.

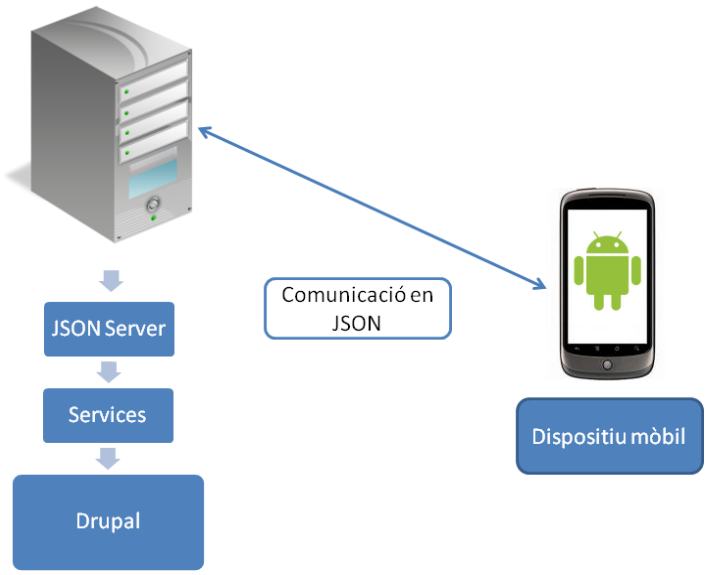
Finalment es va optar per aquesta segona opció ja que permetia disposar de tot l'entorn integrat al Drupal facilitant la migració del portal a una altra màquina així com no haver d'estar pendents de si possibles actualitzacions de Drupal poguessin afectar a un possible sistema creat.

De les opcions que ofereix Drupal per a poder integrar aplicacions externes escollirem el mòdul Services[34], que justament ofereix una solució estandarditzada per integrar aplicacions externes amb Drupal.

El mòdul Services, a part, permet que es pugui treballar amb diferents interfícies alhora de realitzar la comunicació entre aplicacions com poden ser XMLRPC, JSON, JSON-RPC, REST, SOAP, AMF, etc. No entrarem en els detalls de cada opció.

Finalment s'ha decidit utilitzar JSON[35] com a llenguatge d'intercanvi entre aplicacions. Per fer-ho s'utilitzarà el mòdul JSON Server[36], que permetrà utilitzar el mòdul Services amb llenguatge JSON per al intercanvi de dades entre client i servidor. S'ha decidit utilitzar JSON enlloc de, per exemple XML o SOAP, ja que resulta més compacte alleugerint el volum de dades a transferir entre el dispositiu mòbil i el servidor.

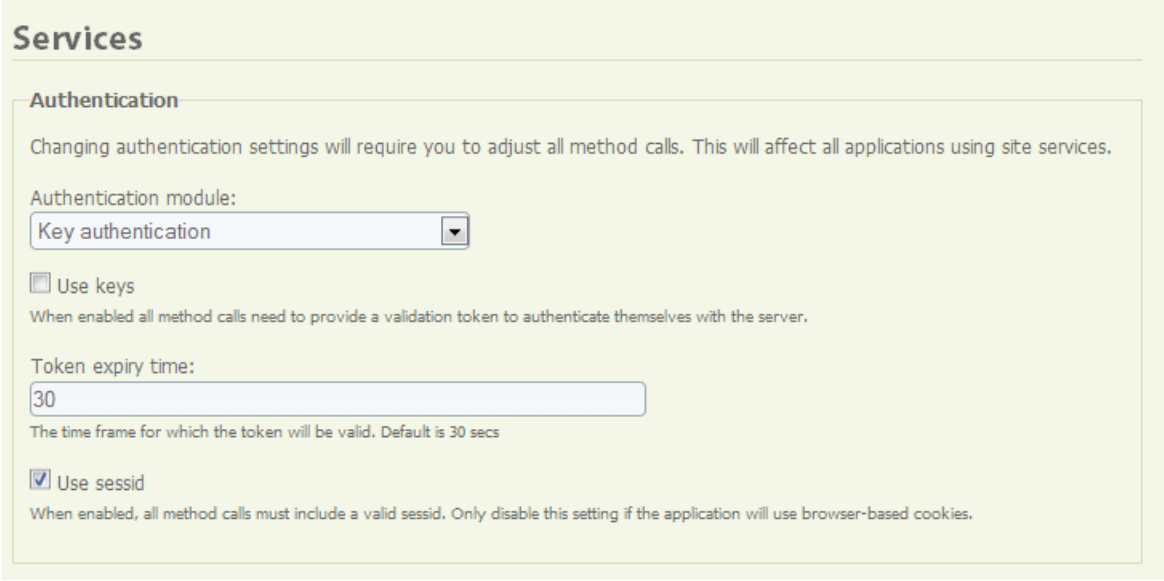
A continuació mostrarem un esquema de l'arquitectura que s'utilitzarà per comunicar la aplicació creada pel dispositiu mòbil basat en Android amb Drupal que ens permetrà entendre millor el funcionament i el paper dels 2 mòduls que acabem d'afegir:



Il·lustració 64 – Esquema de comunicació utilitzant els mòduls JSON Server i Services de Drupal

[34] Més informació del mòdul Services a: http://drupal.org/project/services
[35] Més informació de JSON a: http://www.json.org/
[36] Més informació del mòdul JSON Server a: http://drupal.org/project/json_server

Un cop tenim clara l'arquitectura a utilitzar, realitzarem les configuracions necessàries al mòdul Services per a funcionar amb el servidor que proporciona el mòdul JSON Server (únicament cal activar el mòdul JSON server en la pàgina "admin/build/modules". A dins de la pantalla de configuració indicarem que utilitzarem l'identificador de la sessió a l'hora d'identificar l'usuari que està realitzant les peticions. Aquesta sessió s'haurà de passar en totes les crides al servidor. D'aquesta manera es podrà identificar quin usuari realitza cada petició.



Il·lustració 65 – Configuració del mòdul Services

Un cop tenim l'entorn preparat procedirem a definir els serveis que utilitzarem per a poder gestionar les partides de golf.

Creació dels serveis a utilitzar per aplicacions externes

El mòdul Services ens proporciona alguns mètodes genèrics que ens poden servir per assolir la funcionalitat de gestionar partides de golf. D'altres en canvi els haurem de definir nosaltres degut a què no hi ha cap que s'adapti exactament a les nostres necessitats.

1- **Autenticar un usuari real del sistema**.

   Per poder autenticar-se en Drupal, el mòdul Services ens proporciona els serveis "system.connect" (per crear una sessió al portal) i "user.login" que donat l'identificador de

sessió i les dades d'usuari (nom d'usuari i mot de pas) ens permet autenticar-nos en el sistema.

user.login

Logs in a user.

Arguments (3)

string **sessid** (required)
A valid sessid.
string **username** (required)
A valid username.
string **password** (required)
A valid password.

2- Accedir a la informació dels camps de golf del sistema

Per obtenir els camps de golfs amb tots els seus atributs utilitzarem el servei "views.get" definit pel mòdul Services que ens permet obtenir el resultat d'una vista. Els paràmetres del servei són l'identificador de la sessió, el nom de la vista que volem obtenir i altres paràmetres opcionals que es poden veure en la següent imatge.

views.get

Retrieves a view defined in views.module.

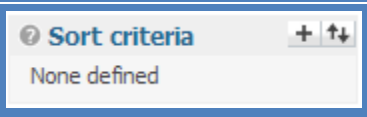
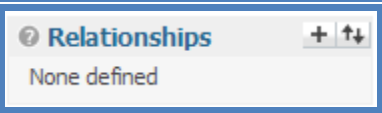
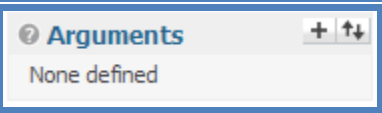
Arguments (7)

string **sessid** (required)
A valid sessid.
string **view_name** (required)
View name.
string **display_id** (optional)
A display provided by the selected view.
array **args** (optional)
An array of arguments to pass to the view.
int **offset** (optional)
An offset integer for paging. If this is set limit will be ignored.
int **limit** (optional)
A limit integer for paging. If offset is set, this will be ignored.
boolean **format_output** (optional)
TRUE if view should be formatted, or only the view result returned (FALSE by default).

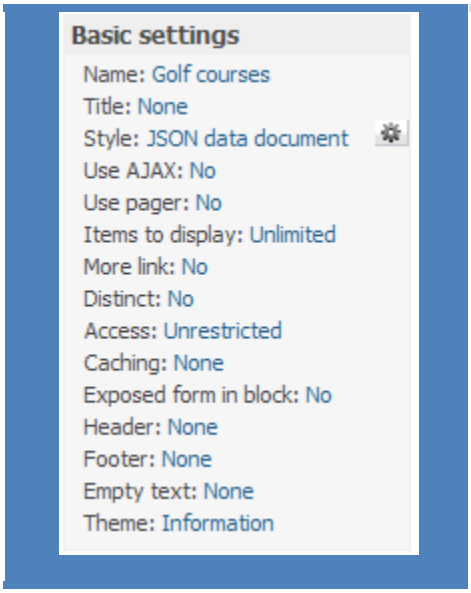
Llavors per a poder mostrar els camps de golf haurem de crear una vista que retorni el llistat de camps de golf en format JSON, ja que és el que utilitzarem per realitzar la

comunicació. Per poder retornar el resultat d'una vista en format JSON utilitzarem el mòdul Views Datasource[37], que afegirà diferents estils de com visualitzar una vista, com poden ser XML o JSON entre d'altres.

A continuació es mostra una taula amb els detalls de configuració de cada bloc de la vista creada per obtenir la informació de tots els camps de golf. La vista s'anomenarà "golf_courses_json".

Bloc de configuració	Configuració
	Els camps que s'han afegir a la vista són: - Identificador numèric del camp de golf - Títol del camp del golf - Par del forat 1 al 18 - Longitud del forat 1 al 18 - Handicap del forat 1 al 18
	No s'ha definit cap criteri d'ordenació.
	S'ha definit un filtre per fer definir que els nodes han de ser únicament de tipus "Camp de golf".
	No s'ha definit cap tipus de relació.
	No s'ha definit cap argument.

[37] Més informació del mòdul Views DataSource a: http://drupal.org/project/views_datasource



En aquest bloc indicarem que l'estil de visualització de la vista serà en JSON gràcies al mòdul Views Datasource. També indicarem que volem retornar tots els elements, en aquest cas camps de golf, en una única consulta.

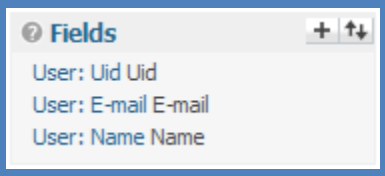
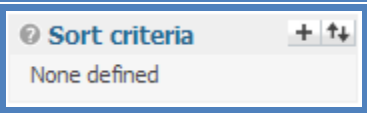
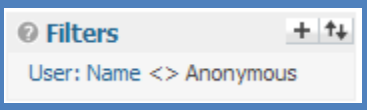
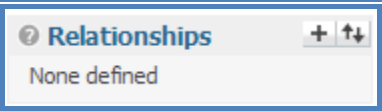
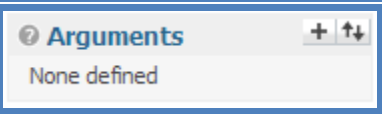
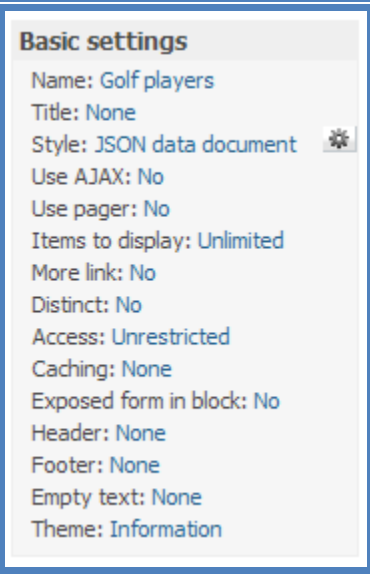
Per entendre millor que significa retornar el resultat de la vista en format JSON adjuntem una petita mostra del resultat d'aquesta vista:


```
{
  "nodes" : [
    {
      "node" : {
        "Nid" : "95",
        "Title" : "Peralada Golf Club",
        "Par1" : "4",
        "Par2" : "4",
        "Par3" : "4",
        "Par4" : "5",
        "Par5" : "4",
        "Par6" : "3",
```


3- Accedir al llistat de possibles jugadors (llistat d'usuaris)

De la mateixa forma que en el punt anterior per retornar el llistat d'usuaris ens ajudarem d'una vista. Aquesta vista, a diferencia de totes les creades fins ara llistarà usuaris enlloc de llistar continguts. Per tant crearem una vista d'usuaris amb el nom de "golf_players".

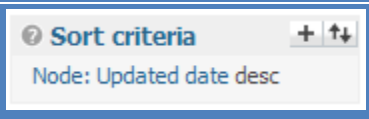
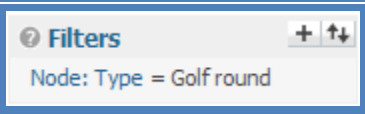
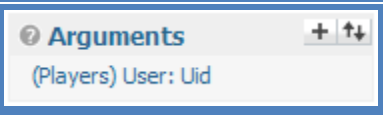
A continuació es mostra la taula amb els detalls de configuració de cada bloc de la vista.

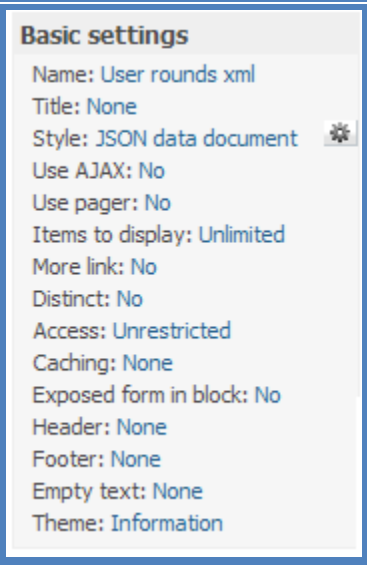
Bloc de configuració	Configuració
	Els camps que s'han afegir a la vista són: - Identificador numèric del usuari - Correu electrònic de l'usuari - Nom de l'usuari en el portal
	No s'ha definit cap criteri d'ordenació.
	S'ha definit un filtre per que l'usuari 0 ("Anonymous") no ha d'aparèixer al llistat. Aquest usuari es creat per Drupal per indicar que l'usuari de la sessió no està autenticat.
	No s'ha definit cap tipus de relació.
	No s'ha definit cap argument.
	En aquest bloc indicarem que l'estil amb que volem visualitzar la vista serà en JSON gracies al mòdul Views Datasource. També indicarem que volem retornar tots els usuaris sense cap tipus de paginació.

4- Accedir a les partides de l'usuari que s'hagi autenticat a l'aplicació.

Per crear aquest servei també crearem una vista que obtindrem a través de “views.get”.

A continuació es mostra la taula amb els detalls de configuració de cada bloc de la vista “user_rounds_json”.

Bloc de configuració	Configuració
	Els camps que s’han afegir a la vista són: - Identificador numèric de la partida de golf - Títol de la partida de golf - Títol del camp del golf - Identificador numèric del camp de golf - Data de creació/modificació de la partida de golf - Per a cadascun dels 4 possibles jugadors de la partida: o Identificador numèric del jugador o Puntuació del jugador a la partida del forat 1 al 18
	S’ha definit que les partides estiguin ordenades per la data de creació/modificació de forma descendent..
	S’ha definit un filtre per fer que els nodes siguin únicament de tipus “Partida de golf”.
	S’han definit dues relacions. Una amb el camp de golf per poder accedir al seu títol i identificador. L’altre relació la fem per obtenir els jugadors de la partida.
	Definirem com a argument l’identificador de l’usuari de la sessió combinant-lo amb la relació creada en el bloc

	anterior amb els jugadors de la partida per quedar-nos només amb les partides on ha participat l'usuari.
	D'igual forma que en les vistes anteriors retornarem tots els resultats sense paginació en format JSON.

5- Permetre actualitzar les puntuacions i jugadors d'una partida i crear noves partides de golf

El mòdul Services ens proporciona el servei "node.save" que ens permet crear i actualitzar continguts. Per fer-ho s'han d'indicar com a paràmetres tots els atributs a guardar del node i, en el cas d'una partida de golf són més de 80 atributs. Això fa que construir la crida sigui molt pesat. Per no haver de fer aquest procés s'ha decidit crear un mòdul que permeti afegir al mòdul Services serveis per crear/modififcar una partida de golf de forma més senzilla. El mòdul l'anomenarem "new_round". Per tant, primer crearem el directori "new_round" dins de "sites/all/modules/custom" on allotjarem els fitxers del mòdul.

Amb la carpeta creada crearem el fitxer "new_round.info" amb la informació bàsica del mòdul. En aquest cas indicarem que depèn del mòdul Services ja que sense aquest mòdul el nostre no funcionaria.

```
;$Id$
name = "New Round Service"
description = "Provides new round services."
core = 6.x
dependencies[] = services
package = PFC Jordi Cabeza
```

En el fitxer “new_round.module” definirem els mètodes que seran accessibles pel mòdul Services. Això ho farem amb el ganxo “hook_services” que crea el mòdul Services permetent poder afegir nous serveis.

Procedirem primer a indicar la definició del servei per crear partides. La següent il·lustració en mostra la definició.

```
// newround.new
array(
    '#method'   => 'newround.new',
    '#callback' => 'new_round_new',
    '#access callback'  => 'new_round_new_access',

    '#return'   => 'struct',
    '#args'     => array(
                    array(
                      '#name'         => 'title',
                      '#type'         => 'string',
                      '#description'  => t('The title of the round.')),
                    array(
                      '#name'         => 'course',
                      '#type'         => 'int',
                      '#description'  => t('The golf course nid.')),
                    array(
                      '#name'         => 'player1',
                      '#type'         => 'string',
                      '#description'  => t('The player1 of the round.')),
                    array(
                      '#name'         => 'player2',
                      '#type'         => 'string',
                      '#description'  => t('The player1 of the round.')),
                    array(
                      '#name'         => 'player3',
                      '#type'         => 'string',
                      '#description'  => t('The player1 of the round.')),
                    array(
                      '#name'         => 'player4',
                      '#type'         => 'string',
                      '#description'  => t('The player1 of the round.'))

                ),
    '#help'     => t('Returns an object containing the nid.')),
```

Il·lustració 66 – Definició del servei per crear partides de golf

Tal com es mostra a la il·lustració s’ha d’indicar el nom del servei, la funció que implementarà la funcionalitat (la definirem a continuació en el mateix fitxer), la funció per determinar si l’usuari té privilegis o no per executar el servei, el tipus d’element que

retornarem (en aquest cas un objecte “struct”), el llistat d’atributs del servei i un missatge informatiu del que retorna la funció.

Els atributs del servei són:

- Títol de la partida
- Identificador del camp de golf
- Jugador1 més la seva puntuació en el següent format: “id_jugador1, puntuació_forat1,..,puntuació_forat18.”
- Jugador2 més la seva puntuació en el següent format: “id_jugador2, puntuació_forat1,..,puntuació_forat18.”
- Jugador3 més la seva puntuació en el següent format: “id_jugador3, puntuació_forat1,..,puntuació_forat18.”
- Jugador4 més la seva puntuació en el següent format: “id_jugador3, puntuació_forat1,..,puntuació_forat18.”

Un cop definit el servei hem de crear les dues funcions definides: la que implementa la funcionalitat del servei i la que permet l’accés al servei.

Definirem primer la funció “new_round_new_access”. Aquesta funció l’implementarem fent ús del ganxo node_access(“operació”,”tipus_de_contingut”) que ens indicarà si tenim permís o no per crear partides de golf.

```
function new_round_new_access($title, $course, $player1, $player2, $player3, $player4) {

  return (node_access('create', "golf_round")) ;
}
```

La funció “new_round_new” que implementa la funcionalitat de crear una partida farà serà:

a. Crear un vector amb els camps que ha de tenir el formulari de creació d’un camp de golf i assignar els valors de cada atribut que hem passat com a paràmetres.

```
// Setup form_state
$edit= array(    "type" => "golf_round" ,
                 "title" => urldecode($title) ,
                 "uid" => "1",
                 "name" => "admin",
                 );

//author
global $user;
$edit['uid'] = $user->uid;
//golf course
$edit['field_golf_field'][0]['nid']['nid']= $course;

//get golf players info
$players['p1']= explode(",", $player1);
$players['p2']= explode(",", $player2);
$players['p3']= explode(",", $player3);
$players['p4']= explode(",", $player4);

//set scorecard and golf players
for ($p=1;$p<=4;$p++){
    if (count($players['p'.$p])==19){ //18 holes + player name => 19 fields
        //set player name
        $edit['field_players'][$p]['uid']['uid'] = $players['p'.$p][0];
        for($h=1;$h<=18;$h++){
            // set hole score
            $edit['field_p'.$p.'h'.$h][0][value]= $players['p'.$p][$h];
        }
    }else{
        break;
    }
}
```

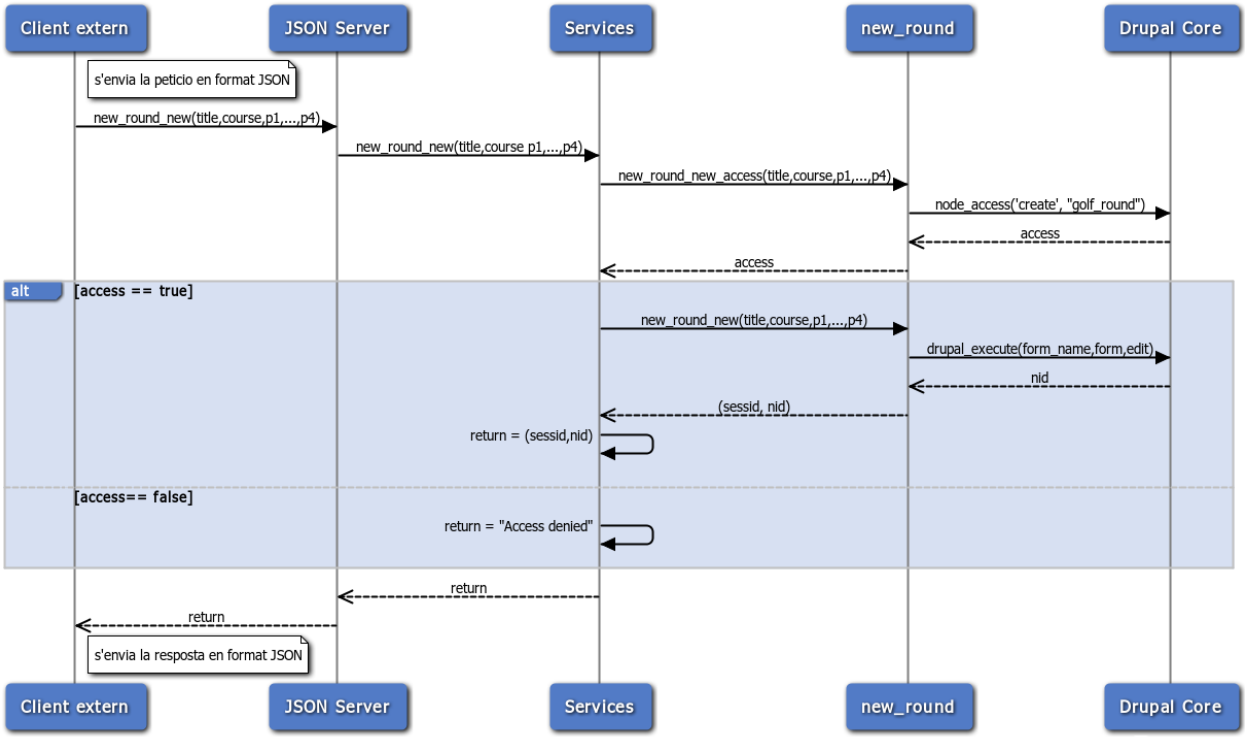
b. Crear el formulari de creació de partides de golf i executar per desar les dades. Equival a omplir per codi el formulari de creació d'una partida de golf i clicar al botó de guardar.

```
//create form object to save the new round
$form_state = array();
$form_state['values'] = $edit;
$form_state['values']['op'] = t('Save');

$ret = drupal_execute('golf_round_node_form', $form_state, (object)$edit);
```

c. Retornar l'identificador del node assignat per Drupal i l'identificador de la sessió en cas d'èxit.

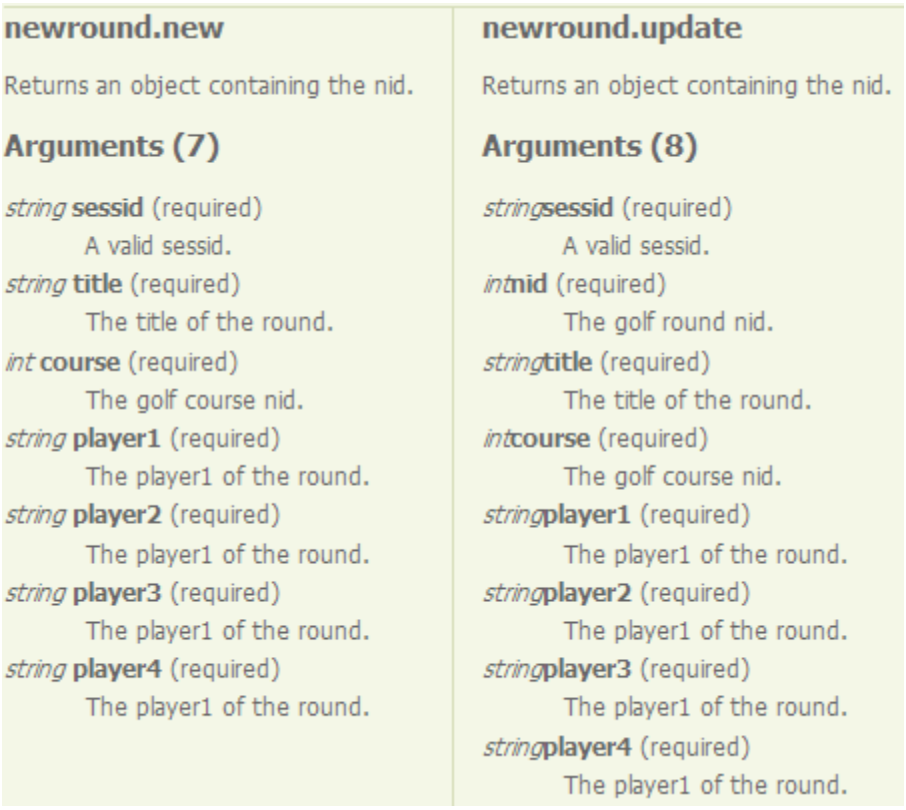
```php
// Fetch $nid out of $form_state
$nid = $form_state['nid'];

if ($errors = form_get_errors()) {
    return services_error(implode("\n", $errors), 401);
}

$return = new stdClass();
$return->sessid = session_id();
$return->nid = $nid;

return $return;
```

El següent diagrama mostra el funcionament del servei per crear partides des de que és cridat per una aplicació externa (client):



Per crear el servei d'actualitzar partides de golf utilitzarem el mateix procediment que per crear partides amb la diferencia que afegirem un paràmetre més: l'identificador del node a actualitzar. El codi de la funció serà exactament igual amb la diferència que també afegirem l'identificador del node de la partida en els camps del formulari.

Un cop creats aquests 2 serveis si activem el mòdul podrem veure com apareixen a la pantalla de configuració i prova del mòdul Services.



newround.new

Returns an object containing the nid.

Arguments (7)

string **sessid** (required)
A valid sessid.
string **title** (required)
The title of the round.
int **course** (required)
The golf course nid.
string **player1** (required)
The player1 of the round.
string **player2** (required)
The player1 of the round.
string **player3** (required)
The player1 of the round.
string **player4** (required)
The player1 of the round.

newround.update

Returns an object containing the nid.

Arguments (8)

*string***sessid** (required)
A valid sessid.
*int***nid** (required)
The golf round nid.
*string***title** (required)
The title of the round.
*int***course** (required)
The golf course nid.
*string***player1** (required)
The player1 of the round.
*string***player2** (required)
The player1 of the round.
*string***player3** (required)
The player1 of the round.
*string***player4** (required)
The player1 of the round.

Amb la creació de tots els serveis necessaris per gestionar partides de golf podem donar per concluida la part servidor i podem començar a centrar-nos a desenvolupar el client, que serà una aplicació per a dispositius mòbils amb sistema operatiu Android.

9 Iteració 6

9.1 Llista d'activitats de la iteració

9.1.1 Estudi/instal·lació plataforma de desonvolupament per a Android

1- Estudiar com funciona el SDK (Software Development Kit) d'Android per a desenvolupar aplicacions.

2- Instal·lació del SDK així com preparar tot l'entorn de desenvolupament.

3- Estudi d'aplicacions i llibreries existents que puguin ser d'ajuda.

9.1.2 Creació esquelet aplicació Android

1- Definir com ha de ser l'aplicació client que servirà per gestionar partides de golf mitjançant wireframes a partir de les funcionalitats ja definides

2- Creació de l'esquelet de l'aplicació (estructura de fitxers, configuracions del projecte, etc) bàsic per a poder començar el desenvolupament.

3- Creació de la pàgina d'inici de l'aplicació

9.1.3 Login usuari

1- Definir un mecanisme per permetre guardar les dades de l'usuari (nom d'usuari i mot de pas).

2- Utilitzar el servei de Drupal per autenticar-se amb les dades guardades de l'usuari.

9.2 Anàlisi i disseny de les funcionalitats

9.2.1 Estudi/instal·lació plataforma de desonvolupament per a Android

En aquest punt iniciarem la creació del client basat en Android que permetrà gestionar les partides de golf del portal creat sobre Drupal.

Per poder crear l'aplicació primer haurem d'instal·lar l'entorn de desenvolupament que consta de:

- Java Development Kit
- Eclipse + Android Development Tools (ADT)
- Android SDK

En la pagina oficial per a desenvolupadors d'Android trobem una guia de fàcil seguiment per instal·lar tot l'entorn en la següent direcció: http://developer.android.com/sdk/index.html.

Amb l'entorn de desenvolupament instal·lat s'ha realitzar un estudi d'aplicacions i llibreries ja creades que ens poden ser útils en el desenvolupament. D'aquest estudi s'ha extret una llibreria que ens facilita la feina de comunicar-nos amb el servidor JSON de Drupal i una aplicació que permet crear targetes de golf.

La llibreria en qüestió s'anomena DrupalCloud[38] i ens permet comunicar-nos amb els serveis de Drupal. En el moment de definit la següent funcionalitat explicarem les modificacions que s'han hagut de realitzar sobre la llibreria per incorporar-la al projecte fent que sigui funcional.

Per altra banda, l'aplicació MinigolfScore permet als seus usuaris crear la puntuació d'una partida de golf i enviar-la per correu electrònic. No permet guardar partides ja que únicament permet disposar d'una única partida. El codi de l'aplicació es troba disponible a http://code.google.com/p/minigolfscore/. D'aquesta aplicació ens serà útil la forma en que es visualitzen les dades d'una partida en forma de targeta de golf, amb scroll horitzontal pels forats, tal com es pot veure a la següent il·lustració.

[38] Més informació sobre la llibreria DrupalCloud a: https://github.com/skyred/DrupalCloud

Il·lustració 67 – Pantalles de l'aplicació MinigolfScore

9.2.2 Creació esquelet aplicació Android

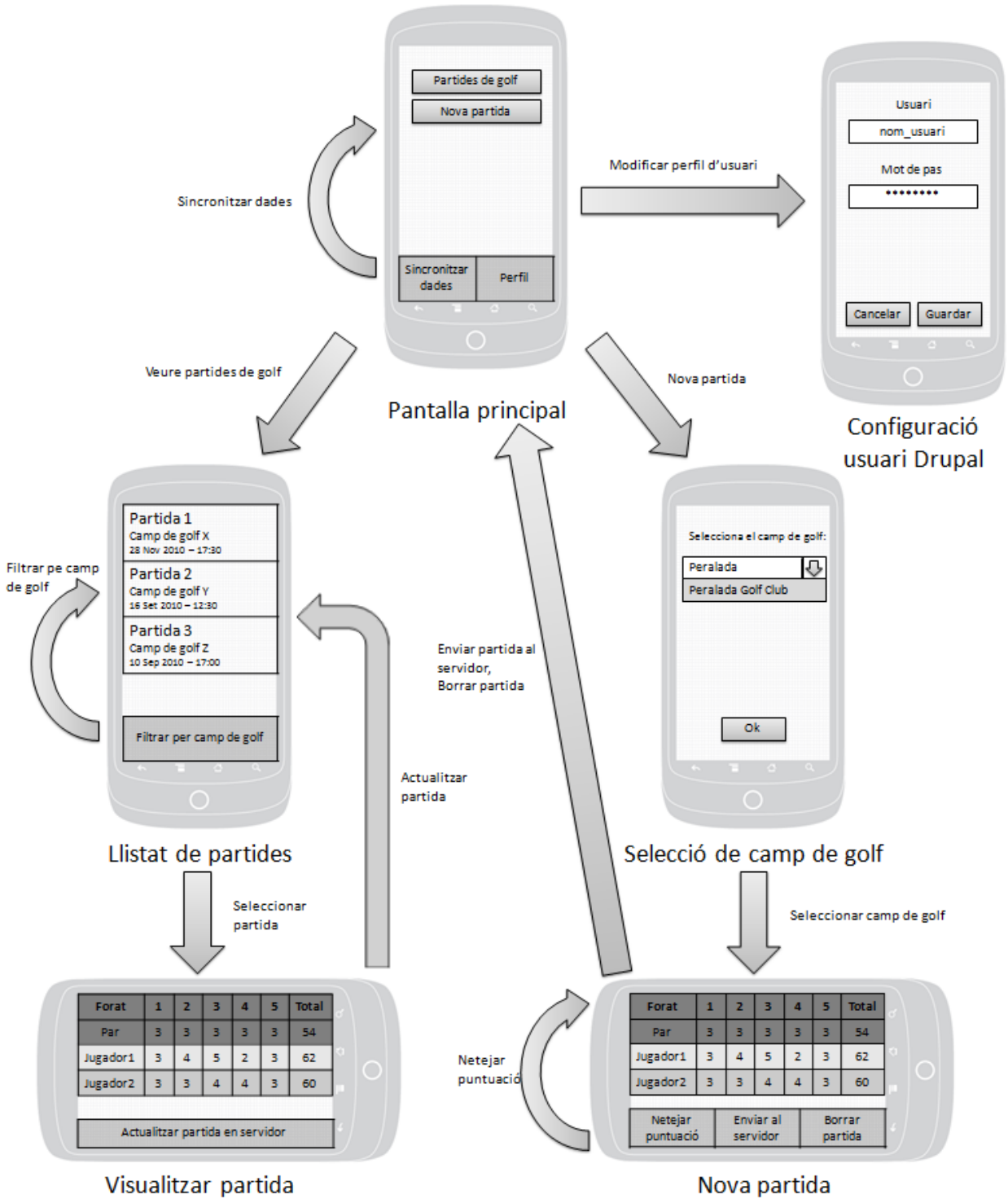
En aquest punt crearem l'estructura bàsica que ens permetrà anar afegint funcionalitats a l'aplicació. També definirem les pantalles que tindrà l'aplicació

Recordem les funcionalitats de l'aplicació:

1. Login/Logout usuari
2. Sincronització de les dades (camps de golf,partides i jugadors)
3. Visualització de partides
4. Actualització de partides existents
5. Creació de noves partides

9.2.2.1 Disseny de l'aplicació mitjançant wireframes

A continuació mostrarem el disseny de l'aplicació mitjançant representacions esquemàtiques (wireframes) de les pantalles i de les interaccions entre elles.



Il·lustració 68 – Maqueta en wireframes de l'aplicació Android

9.2.2.2 Especificació detallada

A continuació farem una explicació detallada de cada pantalla:

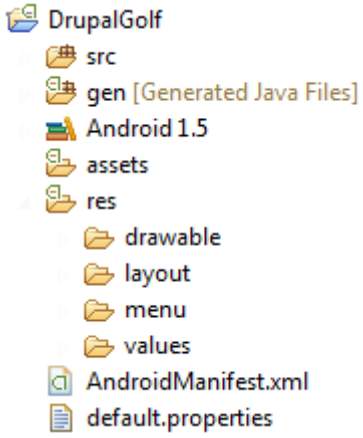
- **Pantalla principal:** La pantalla principal haurà de disposar de dos botons que ens portin a veure el **Llistat de partides** i a crear una nova partida respectivament. Si es vol crear una nova partida ens sortirà la pantalla de **Selecció del camp de golf**. A més a més, el menú de la pantalla (Android permet definir un menú per cada pantalla) ens haurà de permetre obtenir les dades (camps, partides i jugadors) del servidor i anar a la pantalla de **Configuració usuari Drupal** per modificar les dades.
- **Configuració usuari Drupal:** En aquesta pantalla podrem indicar l'usuari i mot de pas que tenim al portal en Drupal. Al modificar les dades tornarem a la **Pantalla principal**.
- **Llistat de partides:** En aquesta pantalla es veuran les partides de l'usuari en forma de llistat. Al seleccionar una partida anirem a la pantalla de **Visualitzar partida**. També, com a opció del menú, es podrà filtrar les partides del llistat per camp de golf.
- **Selecció del camp de golf:** Aquesta pantalla permetrà l'elecció del camp de golf d'una nova partida. Al seleccionar el camp anirem a la pantalla de **Nova partida**.
- **Visualitzar partida:** En aquesta pantalla podrem veure la puntuació d'una partida i modificar-la en cas que es consideri necessari clicant l'opció d'actualitzar del menú. Un cop actualitzada la partida tornarem a la pantalla del **Llistat de partides.**
- **Nova partida:** Pantalla on podrem afegir la puntuació de la nova partida. El menú ens permetrà reiniciar la puntuació, enviar la partida al servidor o esborrar-la. Al enviar la partida o esborrar-la tornarem a la **Pantalla principal**.

Amb les funcionalitats definides i el disseny de les pantalles incloent les possibles navegacions entre pantalles ja podem començar a crear l'aplicació.

9.2.2.3Disseny de la funcionalitat

El primer pas per crear la aplicació es crear el projecte en l'entorn de desenvolupament eclipse. El tipus de projecte serà "Android Project" i escollirem la API Level 3 d'Android, corresponent a la versió 1.5 del sistema operatiu. Això vol dir que l'aplicació serà compatible amb totes les versions d'Android a partir de la 1.5, es a dir, la pràctica totalitat de mòbils amb Android.

El projecte l'anomenarem "DrupalGolf" que serà el nom de l'aplicació. Un cop creat el projecte ens trobarem amb la següent estructura de fitxers que detallarem a continuació.



Il·lustració 69 – Estructura de fitxers d'una aplicació Android

- **src**: A la carpeta "src" trobarem els fitxers .java amb el codi font de l'aplicació.
- **gen**: La carpeta "gen" contindrà el fitxers autogenerats que no s'han de tocar amb definicións de variables que utilitza l'aplicació.
- **assets**: La carpeta "assets" conté fitxers auxiliars de l'aplicació. En el nostre cas restara buida.
- **res**: Aquesta carpeta conté els fitxers amb els recursos necessaris del proejcte: imatges, textos...
    - **res/drawable**: imatges de l'aplicació
    - **res/layout**: fitxers amb la definició de les diferents pantalles
    - **res/menu**: fitxer amb les definicions dels menu de cada pantalla de l'aplicació
    - **res/values**: fitxers XML amb recursos de l'aplicació com, per exemple, cadenes de texte (strings.xml) entre d'altres.
- **AndroidManifest.xml**: Conté la definició en XML dels aspectes principals de l'aplicació: identificació, pantalles, permisos sobre el mòbil, etc.

Un cop definit l'estructura de fitxers ens possarem a crear la pantalla principal de l'aplicació. El primer pas serà definir la pantalla d'inici dins el fitxer AndroidManifest. L'estructura del fitxer AndroidManifest.xml és la següent:

```
<?xml version="1.0" encoding="utf-8"?>
//DEFINICIÓ ASPECTES GENERALS DE L'APLICACIÓ
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.kbza.drupalgolf"
      android:versionCode="1"
      android:versionName="1.0">
      //PERMISOS DE L'APLICACIÖ
       <uses-permission android:name="android.permission.INTERNET"></uses-permission>
 //DEFINICIÓ ACTIVITATS DE L'APLICACIÓ
 <application android:icon="@drawable/icon" android:label="@string/app_name"
android:debuggable="true">
       //DEFINICIÓ ACTIVITAT
        <activity android:name=".DrupalGolf"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
         …
       </application>
</manifest>
```

Els aspectes general són definits automàticament al crear el sistema. En canvi, els permisos que ha de tenir l'aplicació en el sistema operatiu els hi hem d''assignar nosaltres. En el nostre cas, indicarem que volem accedir a Internet ja que sinó no es podrà realitzar la comunicació amb el servidor del mòdul JSONServer definit al Drupal.

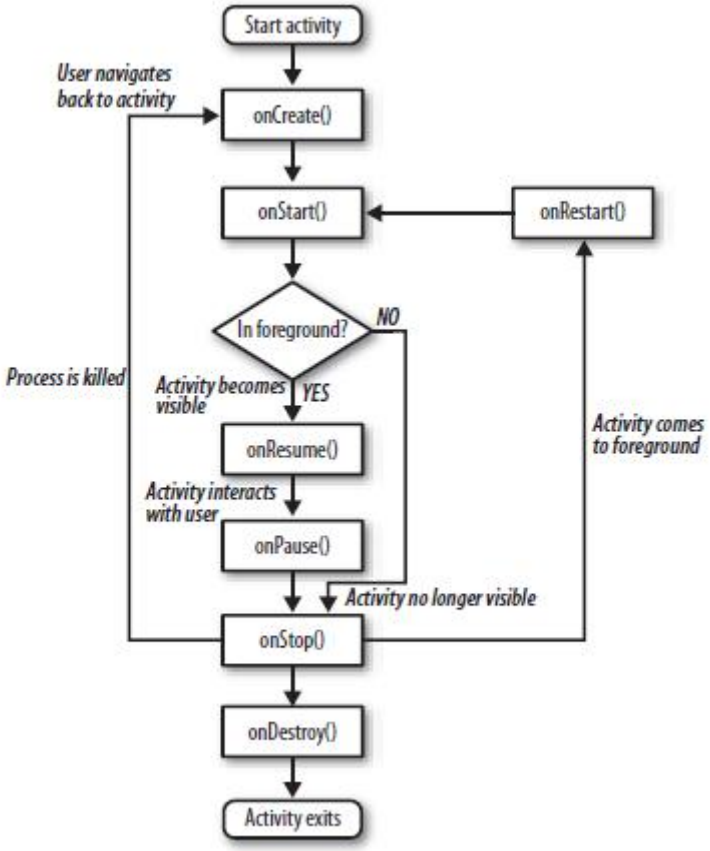
El fitxer AndroidManifest també conté la definició de les activitats en les quals es composa l'aplicació.

En Android una activitat és una peça de codi que apareix i desapareix de l'execució de l'aplicació segons es necessiti. La major part del codi que escrivim en Android es desenvolupa en el context d'una activitat. Generalment, una activitat es correspon amb una pantalla. Es a dir, cada activitat mostra una pantalla a l'usuari.

Les activitats en Android tenen un cicle de vida, es poden crear, pausar, parar, finalitzar, etc... És responsabilitat del programador definir quines accions s'han de crear quan una activitat

canvia d'estat per no perdre informació. Per exemple, si estem creant una partida de golf i parem l'activitat que estem utilitzant es perdran les dades que s'han enregistrat si no definim el codi que desi les dades quan es para una activitat.

A continuació es mostra una il·lustració amb el cicle de vida de les activitats en Android.



Il·lustració 70 – Android Application Development- Cicle de vida de les activitats en Android

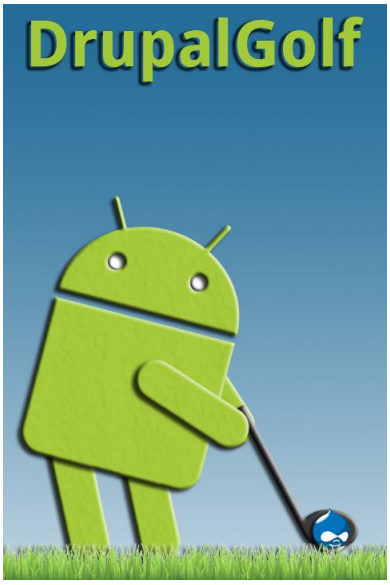
Ara que ja sabem el que és una Activitat, en crearem una que ens servirà per mostrar a l'usuari la pantalla principal. Primer la definirem al fitxer AndroidManifest.xml indicant que és l'activitat principal (MAIN) i la que s'executarà quan es llanci l'aplicació (LAUNCHER):

```
<activity android:name=".DrupalGolf" android:label="@string/app_name">
	<intent-filter>
		<action android:name="android.intent.action.MAIN" />
		<category android:name="android.intent.category.LAUNCHER" />
	</intent-filter>
</activity>
```

El nom de l'activitat (DrupalGolf) ha de coincidir amb el d'un fitxer ".java" que es trobarà dins el package de l'aplicació que s'ha definit en la secció prèvia del manifest. Per tant, crearem el fitxer DrupalGolf.java en la carpeta "src" dins el seu package. Aquest fitxer ampliarà (Extend), com totes les activitats en Android, la classe "Activity".

```
public class DrupalGolf extends Activity {…}
```

El següent pas serà crear el fitxer XML que contindrà el disseny de la pantalla principal, tal com s'ha definit en el disseny mitjançant wireframes anterior. El fitxer l'anomenarem main.xml i l'ubicarem, com totes les definicions de pantalla, a "res/layout". En ell definirem un disseny d'una columna (LineaLayout amb orientació vertical) amb una imatge de fons o en els elements s'aniran situant un a sota de l'altre. Els elements que afegirem seran 2 botons. La imatge de fons la ubicarem a la carpeta "res/drawable". També ubicarem una miniatura d'aquesta imatge que utilitzarem com a icona de l'aplicació amb el nom de "icon.png". Aquesta imatge l'hem creat amb Photoshop i mostra el ninot d'Android jugant a golf utilitzant com a pilota el logotip de Drupal com es mostra a continuació:



Els botons per veure les partides de golf i crear una nova partida l'ubicarem just a sota del títol de l'aplicació de la imatge.

A continuació adjuntem la definició del fitxer main.xml.

```
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:gravity="center_horizontal"
    android:background="@drawable/drupalgolf" android:paddingTop="70dip">
      <Button android:id="@+id/btn_login"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Course cards"
            android:clickable="false" android:minWidth="150sp"
            android:textSize="18sp" android:gravity="center"/>
      <Button android:id="@+id/btn_new_round"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New round" android:gravity="center"
            android:scrollbarAlwaysDrawHorizontalTrack="false"
            android:minWidth="150sp" android:textSize="18sp" />
</LinearLayout>
```

Amb el disseny creat, el següent pas serà afegir el menú de l'activitat, que recordem que eren les opcions de sincronitzar les dades amb el servidor i de modificar les dades d'autenticació de l'usuari en el portal. El fitxer el menú s'ubicarà a dins de "res/menu" amb el nom de main_menu.xml. Aquest fitxer mostrarà les dues opcions del menú. També es defineixen mitjançant xml de la següent forma:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
      <item android:icon="@android:drawable/ic_menu_rotate"
            android:alphabeticShortcut="s" android:title="Synchronize"
android:id="@+id/sync"/>

      <item android:icon="@android:drawable/ic_menu_preferences"
            android:alphabeticShortcut="p" android:title="Profile"
android:id="@+id/profile"/>
</menu>
```

Ara ja podem indicar a l'activitat que utilitzi el disseny que hem definit i que quan l'usuari faci click al botó físic del dispositiu mòbil menú l'aplicació ens mostri les opcions definides en el menú que acabem de crear. Per mostrar el disseny assignarem el "layout" creat amb la funció setContentView(nom_layout) dins de la funció onCreate(), funció que es crida quan es crea l'activitat. Per mostrar el menú, en canvi, utilitzarem la funció onCreateOptionsMenu() on indicarem el nom del menú creat.

```
@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
	setContentView(R.layout.main);
	…
}

public boolean onCreateOptionsMenu(Menu menu) {
	MenuInflater inflater = getMenuInflater();
	inflater.inflate(R.menu.main_menu, menu);
	return true;
}
```

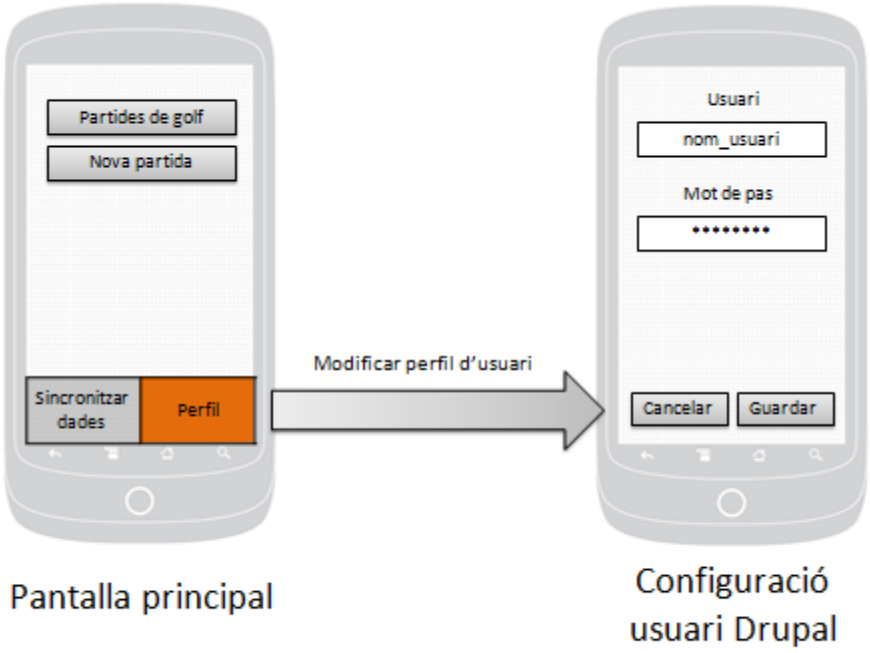
Amb tota aquesta feina realitzada ja podem executar la nostra aplicació que ens mostrarà la pantalla principal amb els elements gràfics, tot i que encara no realitzen cap acció.

Il·lustració 71 – Pantalla principal del client desenvolupat en Android

9.2.3 Login usuari

En aquest punt pretenem afegir al client Android, ja amb l'esquelet de l'aplicació creat, el mecanisme necessari per a poder guardar les dades d'autenticació de l'usuari per que es pugui autenticar en el portal Drupal des del client.

9.2.3.1 Maqueta de la funcionalitat



Il·lustració 72 – Maqueta per guardar les dades d'autenticació de l'usuari

9.2.3.2Especificació detallada

Tal com es mostra a la figura s'haurà de permetre:

- Des de la pantalla principal al clicar en l'opció "Perfil" accedirem a la pantalla on es podran modificar les dades d'autenticació de l'usuari.
- La pantalla de configuració de l'usuari de Drupal haurà de permetre modificar el nom d'usuari i el mot de pas.
- Al clicar el botó de "Guardar" es desaran les dades i es tornarà a la pantalla principal.
- Al clicar el botó de "Cancel·lar" no es desaran les modificacions i també es tornarà a la pantalla principal.

9.2.3.3Disseny de la funcionalitat

Guardar dades d'autenticació de l'usuari

El primer pas serà definir una nova activitat dins el fitxer AndroidManifest per a la pantalla de configuració de l'usuari de Drupal:

```
<activity android:name=".SettingsScreen" android:label="@string/title_settings">
	<intent-filter>
		<action android:name="com.kbza.drupalgolf.SETTINGS" />
	</intent-filter>
</activity>
```

Un cop definida l'activitat crearem el fitxer SettingsScreen.java dins el package del projecte.

Després crearem el fitxer XML que contindrà el disseny de la pantalla de configuració de les dades de l'usuari. El fitxer l'anomenarem settings.xml i l'ubicarem a "res/layout". En ell, a grans trets, definirem un disseny d'una columna (LineaLayout amb orientació vertical) on afegirem el camp pel nom i el mot de pas de l'usuari (s'indicarà l'opció per que el text no sigui visible). Al peu de la pantalla afegirem els botons de cancel·lar i guardar.

A continuació adjuntem el fitxer settings.xml.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:gravity="center_horizontal">
       <LinearLayout android:orientation="vertical"
              android:layout_width="fill_parent" android:layout_height="wrap_content"
              android:gravity="center_horizontal" android:paddingTop="15dip">
              <TextView android:id="@+id/usernameLabel"
               android:layout_width="fill_parent" android:layout_height="wrap_content"
              android:textAppearance="?android:attr/textAppearanceMedium"
android:text="Username"
              android:gravity="center_horizontal"/>
              <EditText android:id ="@+id/username"
                     android:text="@string/username" android:layout_width="wrap_content"
                     android:layout_height="wrap_content" android:minWidth="200sp"/>
       </LinearLayout>
       <LinearLayout android:orientation="vertical"
              android:layout_width="fill_parent" android:layout_height="wrap_content"
              android:gravity="center_horizontal" android:paddingTop="15dip">
              <TextView android:id="@+id/passwordLabel"
               android:layout_width="fill_parent"
                     android:layout_height="wrap_content"
              android:textAppearance="?android:attr/textAppearanceMedium"
              android:text="Password" android:gravity="center_horizontal"/>
              <EditText android:id ="@+id/password"
                     android:text="@string/password"
                     android:layout_width="wrap_content"
                     android:layout_height="wrap_content" android:textStyle="bold"
                     android:password="true" android:minWidth="200sp"/>
       </LinearLayout>
       <TextView android:layout_weight="1"
        android:layout_width="fill_parent" android:layout_height="wrap_content"/>
       <LinearLayout android:orientation="horizontal" android:layout_width="fill_parent"
              android:layout_height="wrap_content" android:gravity="center_horizontal">
           <Button android:id="@+id/cancel_button"
              android:layout_width="100dip" android:layout_height="wrap_content"
              android:textAppearance="?android:attr/textAppearanceMedium"
              android:text="Cancel" />
           <Button android:layout_width="100dip" android:layout_height="wrap_content"
              android:text="Ok" android:id="@+id/ok_button"
              android:textAppearance="?android:attr/textAppearanceMedium" />
       </LinearLayout>
</LinearLayout>
```

Tornant a l'activitat, afegirem el layout creat i definirem els "listeners" per als botons de guardar i cancelar. Aquest "listeners" són funcions que s'executen al ocórrer un esdeveniment en l'element on estan definits. En el nostre cas ens interessa l'esdeveniment onClickListener(), que es dóna quan és clica un element de la pantalla. Aquestes funcions es defineixen a dins de la funció onCreate() quan es crea l'aplicació.

```
final Button buttonOk = (Button) findViewById(R.id.ok_button);
        buttonOk.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
              // Then save the modified settings back
              saveSettings();

              finish();
            }
        });

final Button buttonCancel = (Button) findViewById(R.id.cancel_button);
        buttonCancel.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
              finish();
            }
        });
```

En les funcions es pot veure com quan fem clic al botó de guardar cridarem a la funció saveSettings() que s'encarregarà de desar les dades. En canvi, el botó de cancel·lar al ser clicat finalitzarà l'activitat sense guardar les dades, fent-nos tornar a la pantalla principal.

Per guardar les dades crearem un fitxer d'us intern de l'aplicació on es desarà el nom d'usuari i mot de pas de l'usuari en el portal basat en Drupal, això ho farem en la funció saveSettings() definida a continuació:

```
private void saveSettings() {
//save username and password in DrupalGolf.PROFILE

       SharedPreferences settings =getSharedPreferences(DrupalGolf.PROFILE, 0);
       SharedPreferences.Editor editor = settings.edit();
       editor.putString("username", username.getText().toString());
       editor.putString("password", password.getText().toString());
       // Commit the edits!
       editor.commit();
}
```

També afegirem el codi que permetrà mostrar els valors guardats de l'usuari al iniciar la partida.

```
@Override
public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.settings);

       username = (EditText) findViewById(R.id.username);
       password = (EditText) findViewById(R.id.password);

       SharedPreferences settings = getSharedPreferences(DrupalGolf.PROFILE, 0);
       username.setText(settings.getString("username", "username"));
       password.setText(settings.getString("password", "password"));
```

En aquest punt tenim tot el codi necessari de l'activitat per a poder guardar les dades de l'usuari. Únicament ens falta definir que des de la pantalla principal al clicar el botó "Perfil" de les opcions de menú de la pantalla principal ens porti a l'activitat que acabem de crear.

Llavors, tornarem al fitxer DrupalGolf.java (activitat de la pantalla del menú) i definirem el comportament de l'opció del menú mitjançant la funció onOptionsItemSelected() d'Android. Dins d'aquesta funció si detectem que l'opció clicada al perfil és la de modificar el perfil llançarem una nova activitat amb la funció startActivity().

```
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
       case R.id.profile:
              startActivity(new Intent(this, SettingsScreen.class));
             return true;
       }
       return false;
}
```

Finalment, amb tot el codi creat ja podem guardar les dades d'autenticació de l'usuari

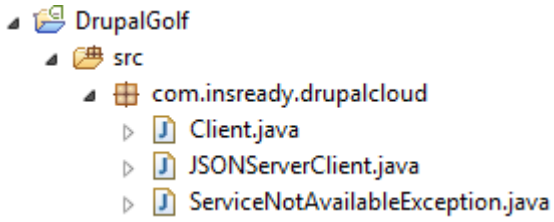
Il·lustració 73 – Accedir a les dades d'usuari a partir de l'opció "Perfil" de la pantalla principal

Recordem que la funcionalitat a assolir és la de poder autenticar un usuari al portal Drupal. Amb la feina realitzada fins ara només hem obtingut les dades per poder ser autenticat. Llavors, ara necessitem definir poder autenticar-nos en el portal Drupal des de l'aplicació Android.

Autenticació de l'usuari a Drupal

El primer que farem serà afegir la llibreria DrupalCloud al nostre projecte. En lloc d'afegir-la com una llibreria Java en format .jar l'afegirem com a part del codi font de l'aplicació respectant el seu package. Ho farem d'aquesta forma degut a que haurem de modificar gran part de la llibreria per adaptar-la a les necessitats del projecte.

La llibreria l'afegirem al projecte a dins de la carpeta "src" tal com mostra la següent figura. A continuació detallarem els fitxers que inclou.



- **Client.java**: Interfície que defineix els mètodes que han de tenir el clients. A part de JSONServerClient.java la llibreria ofereix un client diferent però que no ens resulta útil i, per tant, no l'hem afegit al nostre projecte.
- **JSONServerClient.java**: Classe que implementa els mètodes definits a la classe Client.java i que permet comunicar-nos amb el servidor JSON creat pel mòdul JSON Server de Drupal.
- **ServiceNotAvailableException.java**: Classe per gestionar les possibles excepcions en els mètodes definits a Client.java

La classe **JSONServerClient.java** ha sigut adaptada per treballar només amb l'identificador de la sessió sense utilitzar una clau tal com hem configurat el mòdul Services a Drupal. S'ha deixat comentat en el codi la part que utilitza la clau que Drupal pot assignar per si en un futur es vol implementar. També s'ha modificat com llegeix la resposta ja que la llibreria original només

permet que tot el codi de la resposta estigui en la primera línia del text que es retorna, i en alguns casos Drupal afegeix alguna línea en blanc en el text de resposta fent que la llibreria es pensi que no hi ha cap resposta. A continuació veiem el codi original on es fa un únic readLine de la primera línia de la resposta:

```
InputStream is = response.getEntity().getContent();
BufferedReader br = new BufferedReader(new InputStreamReader(is));
String result = br.readLine();
```

En el nostre projecte hem substituït aquest codi per una altre que permet llegir tot el text de la resposta que ens envia el servidor evitant que es puguin perdre respostes. El codi es troba a continuació:

```
…
String result = readFromBuffer(new BufferedReader(new
InputStreamReader(response.getEntity().getContent(),"UTF-8")));
…

private String readFromBuffer(BufferedReader br){
	StringBuilder text = new StringBuilder();
	try{
		String line;
		while ((line = br.readLine()) != null) {
			text.append(line);
			text.append("\n");
		}
	} catch (IOException e) {
		e.printStackTrace();
		// tratar excepción!!!
	}
	return text.toString();
}
```

Una altra modificació que s'ha tingut que realitzar degut a que el servidor JSONServer espera que els paràmetres de tipus String al cridar un servei estiguin entre cometes "". Per tant, hem d'afegir les cometes en els paràmetres necessaris. Per exemple, per cridar al servei des del mòdul Services de Drupal user.login().

```
@Override
public String userLogin(String username, String password)
        throws ServiceNotAvailableException {
        BasicNameValuePair[] parameters = new BasicNameValuePair[2];
        parameters[0] = new BasicNameValuePair("username", "\""+username+"\"");
        parameters[1] = new BasicNameValuePair("password", "\""+password+"\"");
        return call("user.login", parameters);
}
```

En el fitxer “res/values/stings.xml” s’han afegit les següents variables que seran utilitzades per JSONServerClient.java:

```
<string name="sharedpreferences_name">UserAuth</string>
<string name="SERVER">
        http://domini_drupal/services/json
</string>
<string name="API_KEY"></string>
<string name="DOMAIN">android</string>
<string name="ALGORITHM">HmacSHA256</string>
<string name="SESSION.LIFETIME">1209600</string>
```

La variable més important és “SERVER” on s’ha d’indicar la URL del nostre portal en Drupal en la que el servidor JSON Server es troba escoltant les peticions.

Amb la llibreria DrupalCloud afegida , ja poden realitzar la funció d’autenticació. Per utilitzar les funcions de la llibreria DrupalCloud.java crearem una classe que ens farà de pont i ens permetrà simplificar el codi de les nostres activitats. Aquesta classe l’anomenarem Utils.java i és on definirem tots els mètodes que s’hagin d’utilitzar en les diferents activitats.

A continuació s’indica el constructor de la classe Utils que alhora instanciarà el client per comunicar-nos amb Drupal.

```
public Utils(Context ctx) {
        this.ctx = ctx;
        client = new JSONServerClient(ctx,
                        ctx.getString(R.string.sharedpreferences_name),
                        ctx.getString(R.string.SERVER), ctx.getString(R.string.API_KEY),
                        ctx.getString(R.string.DOMAIN),
                        ctx.getString(R.string.ALGORITHM),
                        Long.parseLong(ctx.getString(R.string.SESSION_LIFETIME)));
}
```

La variable ctx de tipus Context conté el context de l'activitat, gràcies al qual podem accedir als valors guardats en el fitxer "strings.xml" amb el mètode `getString()` de la classe Context d'Android.

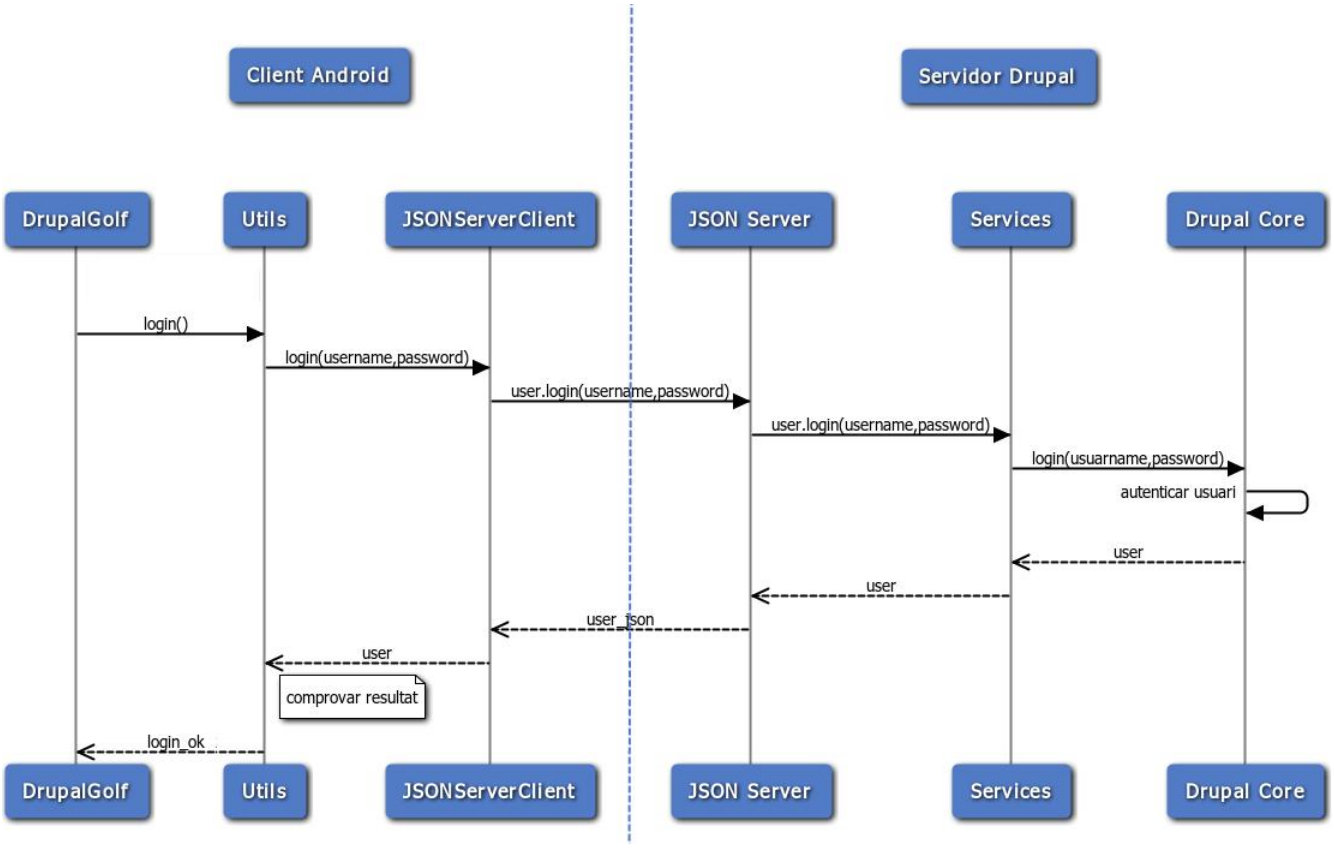
Per finalitzar la funcionalitat crearem la funció login() que utilitzarà els mètodes proporcionats per la llibrera modificada DrupalCloud per autenticar-se en el sistema.

```
/**
 * Login in drupal portal
 * @param username
 * @param password
 * @return
 */
public boolean login(){
boolean res= false;

String login= null;
	try {
		SharedPreferences settings =
					ctx.getSharedPreferences(DrupalGolf.PROFILE, 0);

		login=client.userLogin(settings.getString("username", "username"),
					settings.getString("password", "password"));
		if (login!=null) res= true;

	} catch (ServiceNotAvailableException e) {
		e.printStackTrace();
	}

	return res;
}
```

Amb aquesta funció creada si volem fer login des de qualsevol activitat del projecte només haurem d'escriure el següent codi.

```
Utils u = new Utils(NomActivitat.this);
u.login();
```

A continuació indiquem el diagrama de seqüència que ens permet entendre com es comunica l'aplicació DrupalGolf que estem creant amb el portal Drupal. Utilitzarem la funció login() que ens permet autenticar-nos com a exemple. Per a la resta de funcionalitats el procés serà idèntic. Únicament canviarà el nom de les funcions i els seus paràmetres. En el diagrama es pot veure clarament diferenciada per la línia de punts la part client(Android) de la part servidor(Drupal).



Amb aquesta funció creada si volem fer login des de qualsevol activitat del projecte només haurem d'escriure el següent codi.

10 Iteració 7

10.1 Llista d'activitats de la iteració

10.1.1 Sincronització de les dades del servidor

1- Obtenir des del client en Android els camps de golf, els jugadors i les partides de golf de l'usuari .
2- Guardar la informació obtinguda del servidor de forma que pugui ser accessible en un futur.
3- Permetre obtenir la informació d'un únic objecte. Es a dir, definir mètodes per obtenir la informació d'un jugador, d'un camp de golf o d'una partida de golf.

10.1.2 Visualització de partides

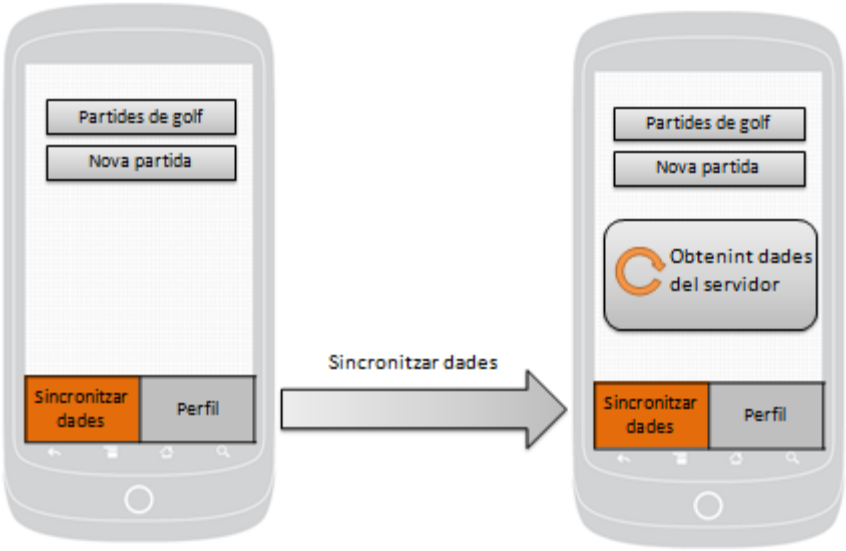
1- Obtenir les partides guardades de l'usuari.
2- Mostrar un llistat de les partides de l'usuari.
3- Permetre veure la puntuació d'una partida en concret.

10.2 Anàlisi i disseny de les funcionalitats

10.2.1 Sincronització de les dades del servidor

En aquest punt pretenem obtenir tota la informació que necessita el client en Android per a poder gestionar les partides de golf. Un cop obtinguda la informació haurem de guardar-la perquè pugui ser utilitzada per la resta de l'aplicació.

10.2.1.1 Maqueta de la funcionalitat



Il·lustració 74 – Maqueta de la sincronització de dades amb el servidor

10.2.1.2 Especificació detallada

A continuació farem una explicació detallada de com s'ha de comportar la funcionalitat:

- Al clicar l'opció del menú "Sincronitzar dades" de la pantalla principal el sistema començarà a obtenir les dades del servidor.
- Durant la realització del procés es mostrarà una diàleg a l'usuari indicant que l'aplicació s'està comunicant amb el servidor.
- Quan s'hagi obtingut i guardat tota la informació es tancarà el diàleg tornant a tenir el control de l'aplicació des de la pantalla principal

10.2.1.3 Disseny de la funcionalitat

Per assolir aquesta funcionalitat dividirem la feina en quatre tasques: obtenir camps de golf, obtenir jugadors de golf, obtenir partides de golf i realitzar l'obtenció des de l'activitat DrupalGolf (pantalla principal).

A continuació detallarem la realització de cadascuna de les quatres tasques.

Obtenir camps de golf

Per obtenir els camps de golf haurem de cridar al servei views.get del mòdul Services de Drupal indicant com a nom de la vista "golf_courses_json".

A la llibreria JSONServerClient.java disposem d'un mètode per utilitzar el serveis views.get però crearem una nova funció que ens servirà per obtenir aquesta vista en concret sense haver de preocupar-nos de tenir que passar cap paràmetre.

Primer definirem la funció a la interfície Client.java per després implementar-la en JSONServerClient. A continuació mostrem la implementació de la nova funció golf_courses() que ens permetrà obtenir el resultat de la vista "golf_courses_json".

```
@Override
public String golfCourses() throws ServiceNotAvailableException{
	BasicNameValuePair[] parameters = new BasicNameValuePair[6];
	parameters[0] = new BasicNameValuePair("view_name", "\"golf_courses_json\"");
	parameters[1] = new BasicNameValuePair("args", "");
	parameters[2] = new BasicNameValuePair("display_id", "");
	parameters[3] = new BasicNameValuePair("offset", "");
	parameters[4] = new BasicNameValuePair("limit", "");
	parameters[5] = new BasicNameValuePair("format_output", "TRUE");
	return call("views.get", parameters);

}
```

Aquest mètode ens retorna un string amb el llistat de camps de golf en format JSON. El següent pas a realitzar es definir una funció a la classe Utils.java que utilitzarà la funció golfCourses() que acabem de definir i guardarà el resultat en format JSON per poder oferir la informació en un futur.

La funció l'anomenarem també golfCourses() i el codi que la implementa la trobem a continuació:

```
/**
* Save the courses in the file golf_courses
 * @return
 */
public String golfCourses(){
	String res = null;
	try {
		res=client.golfCourses();
		JSONObject json = new JSONObject(res);
		FileOutputStream fos = ctx.openFileOutput(DrupalGolf.GOLF_COURSES,
			Context.MODE_PRIVATE);
		fos.write(json.toString().getBytes());
		fos.close();

	} catch (JSONException e) {
		e.printStackTrace();
	} catch (ServiceNotAvailableException e) {
		e.printStackTrace();
	}catch (IOException e) {
		e.printStackTrace();
	}
	return res;
}
```

Com es pot veure en el codi per crear el fitxer on desar la informació s'utilitza la funció openFileOutput(). En aquesta funció podem indicar que el fitxer creat únicament serà accessible per la nostra aplicació i no per la resta d'aplicacions que puguem tenir instal·lades al dispositiu mòbil. Això ho aconseguim indicant "MODE_PRIVATE" com a paràmetre de la funció.

Amb aquest codi ja podem obtenir la informació dels camps de golf i guardar-la en un fitxer per a poder processar-lo posteriorment.

Per treballar de forma més senzilla en tota l'aplicació crearen un classe que ens servirà per encapsular la informació d'un camp de golf ja que ens serà més còmode de manipular que una cadena de text en format JSON. Per tant, crearem la clase "Course" en el fitxer Course.java. Aquest fitxer l'ubicarem en una carpeta anomenada "objects" dins el package de l'aplicació. En aquest directori anirem guardant tots els objectes definits.

La definició de l'objecte "Course" serà la següent:

```
public class Course {
   private String cid; //Identificador numeric del camp.
   private String name; //Nom del camp de golf
   private int[] Par; // Par de cada forat
   private int[] Handicap; // Handicap de cada forat
   private int[] Length; // Longitud de cada forat


…


}
```

Un cop tenim l'objecte definit crearem la funció getCourse(String cid) que donat l'identificador numèric del node que representa el camp de golf ens retornarà un objecte de tipus "Course" amb tota la informació del camp.

```
public Course getCourse(String cid){
       Course c=null;
       try {
              FileInputStream fis = ctx.openFileInput(DrupalGolf.GOLF_COURSES);

              BufferedReader isr = new BufferedReader(new InputStreamReader(fis));

              JSONArray courses =
                           (new JSONObject(isr.readLine())).getJSONArray("nodes");
              for(int i=0;i<courses.length();i++){
                     JSONObject course =
                                  courses.getJSONObject(i).getJSONObject("node");
                     if (cid.equals(course.getString("Nid"))){
                           c= new Course (course.getString("Nid"),
                                        course.getString("Title"));
                           for(i=1;i<=DrupalGolf.NUM_HOLES;i++){
                                  c.setPar(i-1, course.getInt("Par"+i));
                                  c.setHandicap(i-1, course.getInt("handicap"+i));
                                  c.setLength(i-1, course.getInt("length"+i));
                           }
                     }
              }

              fis.close();
       } catch (FileNotFoundException e1) {
              e1.printStackTrace();
       } catch (IOException e) {
              e.printStackTrace();
       } catch (JSONException e) {
              e.printStackTrace();
       }
       return c;
}
```

Obtenir jugadors de golf

Per obtenir els possibles jugadors d'una partida de golf haurem de cridar al servei views.get del mòdul Services de Drupal indicant com a nom de la vista "golf_players".

D'igual forma que per als camps de golf crearem una funció a la llibreria DrupalCloud per obtenir directament aquesta vista sense haver d'indicar cap paràmetre.

Primer definirem la funció a la interfície Client.java per després implementar-la en JSONServerClient. A continuació mostrem la implementació de la funció creada, de nom golf_players(), que ens permetrà obtenir el resultat de la vista "golf_players".

```
@Override
public String golfPlayers() throws ServiceNotAvailableException{
	BasicNameValuePair[] parameters = new BasicNameValuePair[6];
	parameters[0] = new BasicNameValuePair("view_name", "\"golf_players\"");
	parameters[1] = new BasicNameValuePair("args", "");
	parameters[2] = new BasicNameValuePair("display_id", "");
	parameters[3] = new BasicNameValuePair("offset", "");
	parameters[4] = new BasicNameValuePair("limit", "");
	parameters[5] = new BasicNameValuePair("format_output", "TRUE");
	return call("views.get", parameters);

}
```

Aquest mètode ens retornarà el llistat d'usuaris del portal en una cadena text en format JSON. El següent pas a realitzar es definir una funció a la classe Utils.java que utilitzarà la funció golfPlayers(), que tot just acabem de definir, i guardarà en un fitxer el resultat en format JSON per poder oferir la informació sobre els jugadors en un futur.

La funció l'anomenarem també golfPlayers() i estarà implementada pel següent codi:

```
/**
 * Save the players in the file golf_players
 * @return
 */
public String golfPlayers(){
	String res = null;
	try {
		res=client.golfPlayers();
		JSONObject json = new JSONObject(res);
		FileOutputStream fos = ctx.openFileOutput(DrupalGolf.GOLF_PLAYERS,
					Context.MODE_PRIVATE);
		fos.write(json.toString().getBytes());
		fos.close();

	} catch (JSONException e) {
		e.printStackTrace();
	} catch (ServiceNotAvailableException e) {
		e.printStackTrace();
	}catch (IOException e) {
		e.printStackTrace();
	}
	return res;
}
```

Ara ja podem obtenir del servidor el llistat de jugadors i guardar-lo en un fitxer per a poder processar-lo posteriorment.

Com en el cas dels camps de golf també crearem un objecte on encapsular la informació que obtenim de cada jugador. Per tant, crearem la classe "Player" en el fitxer Player.java.
L'ubicarem al directori "objects" del nostre package.

La definició de l'objecte "Player" serà la següent:

```
package com.kbza.drupalgolf.objects;

public class Player {
   private String uid;
   private String email;
   private String name;


…

`
```

Un cop tenim l'objecte definit crearem les funcions getPlayer(String uid) i getPlayerByName(String name) que ens permetran obtenir on objecte de tipus "Player" amb tota la informació del jugador donat l'identificador numèric del jugador o el seu nom d'usuari respectivament.

```
public Player getPlayer(String uid){
	Player p=null;
	try {
		FileInputStream fis = ctx.openFileInput(DrupalGolf.GOLF_PLAYERS);

		BufferedReader isr = new BufferedReader(new InputStreamReader(fis));

		JSONArray players = (
				new JSONObject(isr.readLine())).getJSONArray("nodes");
		for(int i=0;i<players.length();i++){
			JSONObject player =
					players.getJSONObject(i).getJSONObject("node");
			if (uid.equals(player.getString("Uid"))){
				p= new Player( player.getString("Uid"),
				player.getString("E-mail"),player.getString("Name"));
			}
		}
		fis.close();
	} catch (FileNotFoundException e1) {
		// TODO Auto-generated catch block
		e1.printStackTrace();
	} catch (IOException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	} catch (JSONException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
	return p;
}
```

```
public Player getPlayerByName(String name){
	Player p=null;
	try {
		FileInputStream fis = ctx.openFileInput(DrupalGolf.GOLF_PLAYERS);

		BufferedReader isr = new BufferedReader(new InputStreamReader(fis));

		JSONArray players = (new
JSONObject(isr.readLine())).getJSONArray("nodes");
		for(int i=0;i<players.length();i++){
			JSONObject player
=players.getJSONObject(i).getJSONObject("node");
			if (name.equals(player.getString("Name"))){
				p= new Player
(player.getString("Uid"),player.getString("E-mail"),player.getString("Name"));
			}
		}
		fis.close();
	} catch (FileNotFoundException e1) {
		e1.printStackTrace();
	} catch (IOException e) {
		e.printStackTrace();
	} catch (JSONException e) {
		e.printStackTrace();
	}
	return p;
}
```

Obtenir partides de golf

Per obtenir les partides de golf de l'usuari haurem de cridar al servei views.get del mòdul Services de Drupal indicant com a nom de la vista "user_rounds_json".

D'igual forma que en els casos anteriors crearem una nova funció a la llibreria DrupalCloud per obtenir directament aquesta vista sense haver d'indicar cap paràmetre.

Primer definirem la funció a la interfície Client.java per després implementar-la en JSONServerClient. A continuació mostrem la implementació de la funció creada, de nom user_rounds(), que ens permetrà obtenir el resultat de la vista "user_rounds_json".

```
@Override
public String userRounds() throws ServiceNotAvailableException{
	BasicNameValuePair[] parameters = new BasicNameValuePair[6];
	parameters[0] = new BasicNameValuePair("view_name", "\"user_rounds_json\"");
	parameters[1] = new BasicNameValuePair("args", "");
	parameters[2] = new BasicNameValuePair("display_id", "");
	parameters[3] = new BasicNameValuePair("offset", "");
	parameters[4] = new BasicNameValuePair("limit", "");
	parameters[5] = new BasicNameValuePair("format_output", "TRUE");
	return call("views.get", parameters);

}
```

El següent pas a realitzar es definir una funció a la classe Utils.java que utilitzarà la funció userRounds(), que tot just acabem de definir, i guardarem en un fitxer el resultat en format JSON per poder accedir a la informació de les partides de l'usuari en un futur.

La funció l'anomenarem també userRounds() i estarà implementada pel següent codi:

```
/**
 * Save the rounds in the file golf_rounds
 * @return
 */
public String userRounds(){
	String res = null;
	try {
		res=client.userRounds();
		JSONObject json = new JSONObject(res);

		FileOutputStream fos = ctx.openFileOutput(DrupalGolf.GOLF_ROUNDS,
					Context.MODE_PRIVATE);
		fos.write(json.toString().getBytes());
		fos.close();

	} catch (JSONException e) {
		e.printStackTrace();
	} catch (ServiceNotAvailableException e) {
		e.printStackTrace();
	}catch (IOException e) {
		e.printStackTrace();
	}

	return res;

}
```

Ara ja podem obtenir del servidor el llistat de partides de l'usuari per guardar-lo en un fitxer que serà accessible en un futur per les diferents activitats de l'aplicació.

Com en els casos anteriors també crearem un objecte on encapsular la informació que obtenim de cada partida, sense tenir en compte la puntuació. Per tant, crearem la classe "GolfRound" en el fitxer GolfRound.java.

La definició de l'objecte "GolfRound" serà la següent:

```
package com.kbza.drupalgolf.objects;

public class GolfRound {
	private String Nid;
	private String Title;
	private String golf_course;
	private String golf_course_id;
	private String date;…


}
```

Sincronització de dades

Un cop tenim les tres funcions per obtenir les dades del servidor podem anar a l'activitat DrupalGolf (pantalla principal) on afegir el codi necessari.

El que haurem de fer serà definir el cas en que es prem l'opció del menú "Sincronitzar dades" en la funció onOptionsItemSelected(). En el cas de que premi l'opció s'haurà de mostrar un diàleg a l'usuari indicant que s'estan obtenint les dades que es tancarà quan s'hagi finalitzat l'obtenció. Per fer-ho ens ajudarem de la classe ProgresDialog[39], que ens permet definir un diàleg de progrés indicant un missatge mentre s'està executant el codi. A continuació podem veure el codi:

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.sync:
       final ProgressDialog dialog = ProgressDialog.show(this, "Synchronizing",
                    "Getting players, courses and rounds from server.", true);
                    final Handler handler = new Handler() {
                       public void handleMessage(Message msg) {
                          dialog.dismiss();
                          }
                       };
                    Thread checkUpdate = new Thread() {
                           public void run() {
                                  Utils u = new Utils(DrupalGolf.this);
                                  u.login();
                                  u.golfPlayers();
                                  u.golfCourses();
                                  u.userRounds();
                                  handler.sendEmptyMessage(0);
                           }
                    };
                    checkUpdate.start();

        return true;
…
}
```

A continuació mostrem com queda la funcionalitat en el dispositiu mòbil:

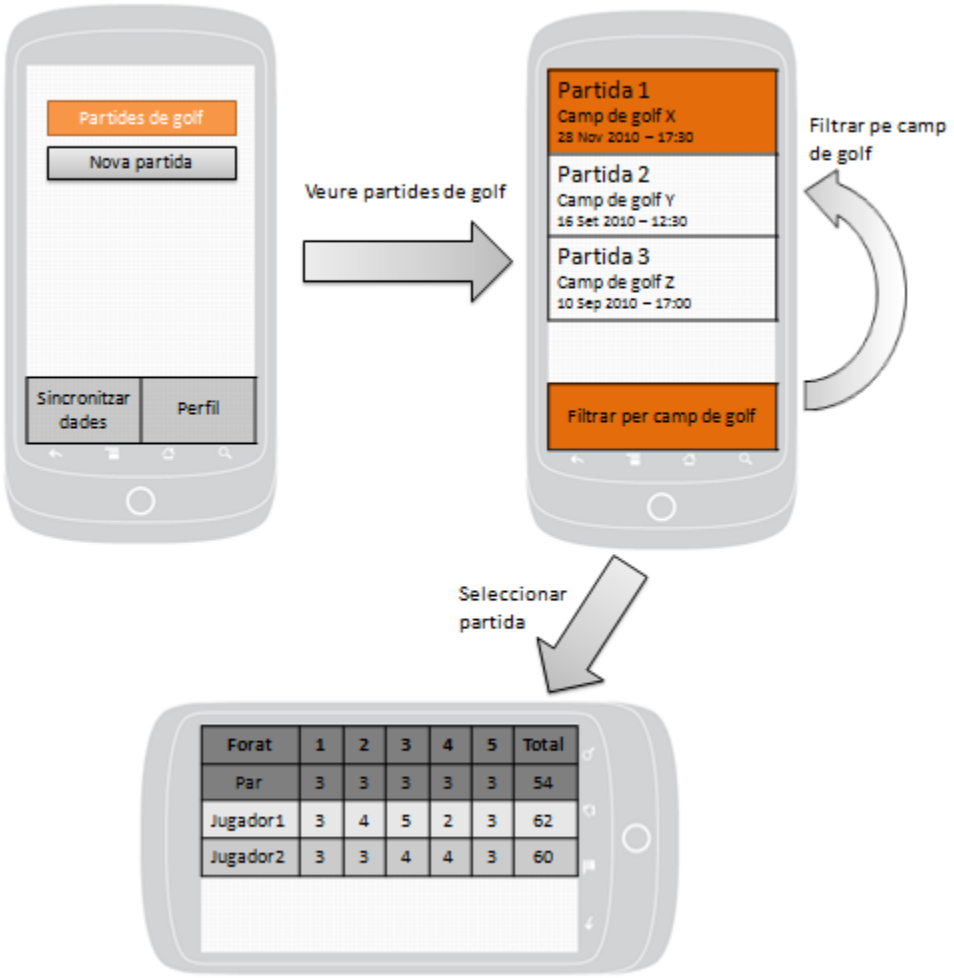
[39] Més informació a: http://developer.android.com/reference/android/app/ProgressDialog.html

Il·lustració 75 – Pantalla principal executant l'opció de sincronització de dades

10.2.2 Visualització de partides

En aquesta funcionalitat volem permetre a l'usuari poder accedir al llistat de les seves partides de golf i poder visualitzar el resultat d'una partida en detall.

10.2.2.1 Maqueta de la funcionalitat



Il·lustració 76 – Maqueta que mostra el procés per visualitzar una partida

10.2.2.2 Especificació detallada

Tal com es mostra a la figura s'haurà de permetre:

- Des de la pantalla principal al prémer el botó "Partides de golf" accedirem a la pantalla amb el llistat de partides de golf de l'usuari.
- El llistat de partides haurà d'informar per cada partida de:
    - Títol de la partida
    - Nom del camp de golf
    - Data de la partida
- El llistat de partides s'ha de poder filtrar pel camp de golf.

- Cada element del llistat s'ha de poder seleccionar. Al seleccionar l'element anirem a la pantalla que ens mostrarà la puntuació de la partida amb la informació del camp.

10.2.2.3 Disseny de la funcionalitat

Aquesta funcionalitat es pot dividir en dos tasques principals: crear el llistat de partides i mostrar el resultat d'una partida.

Crear el llistat de partides

El primer pas serà definir una nova activitat dins el fitxer AndroidManifest per a la pantalla de configuració de l'usuari de Drupal:

```
<activity android:name=".ListRounds"
          android:label="@string/app_name">
    <intent-filter>
        <category android:name="com.kbza.drupalgolf.LIST_ROUNDS" />
    </intent-filter>
</activity>
```

Un cop definida l'activitat crearem el fitxer ListRounds.java dins el package del projecte.

Després crearem el fitxer XML que contindrà el disseny de la pantalla amb el llistat de partides. El fitxer l'anomenrem list_rounds.xml i l'ubicarem a "res/layout". En el fitxer definirem un disseny d'una columna amb un únic element que serà la vista (ListView).

A continuació adjuntem el fitxer list_rounds.xml.

```
<LinearLayout android:id="@+id/LinearLayout01"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 xmlns:android="http://schemas.android.com/apk/res/android">
      <ListView android:id="@+id/ListView01"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content" />
</LinearLayout>
```

Aquest disseny únicament indica que la pantalla tindrà un llistat. El disseny de cada element del llistat el definirem al fitxer list_rounds_item.xml que també l'ubicarem a "res/layout". En

aquest segon crearem tres elements de tipus text on ficarem el títol de la partida, el nom del camp de golf i la data de la partida respectivament.

A la carpeta "res/menu" crearem el menú de la pantalla del llistat que únicament ha de contenir l'opció de filtrar per camp de golf. El fitxer l'anomenarem list_rounds_menu.xml.

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
      <item android:title="Filter by course" android:id="@+id/course_filter"
            android:icon="@android:drawable/ic_menu_search"
            android:alphabeticShortcut="f" />
</menu>
```

El següent pas serà crear dos funcions al fitxer Utils.java que ens permeti obtenir totes les partides i les partides d'un determinat camp en un llistat d'objectes de tipus "GolfRound". Les funcions les anomenarem getRounds i getRoundsByCourse respectivament. A continuació indiquem part del seu codi.

```
/**
 * @return ArrayList of User Rounds
 */
public ArrayList<GolfRound> getRounds(){
      ArrayList<GolfRound> rounds= new ArrayList<GolfRound>();
      try {
            …
            for(int i=0;i<rounds_json.length();i++){
                  …
                  rounds.add(round);
            }
            fis.close();
      } catch …}
      return rounds;
}
/**
 * @return ArrayList of User Rounds filter by Course cid
 */
public ArrayList<GolfRound> getRoundsbyCourse(String cid){
      ArrayList<GolfRound> rounds= new ArrayList<GolfRound>();
      try {
            …
            for(int i=0;i<rounds_json.length();i++){
                  …
                  if(cid.equals(round.getGolf_course_id())) rounds.add(round);
            }
            fis.close();
      } catch …}
      return rounds;
}
```

Per poder crear un llistat d'objectes que hem definit nosaltres, objecte "GolfRound", hem de definir un adaptador que ens permetrà visualitzar cada element del llistat que retornen les funcions que acabem de definir. A grans trets volem que cada element del llistat es mostri amb el disseny que hem definit a list_rounds_menu.xml.

Aquest adaptador estarà al fitxer RoundsAdapter.java i a partir del llistat de partides crearà els elements del llistat assignant el títol de la partida, el nom del camp de golf i la data. Mostrem una porció del fitxer RoundsAdapter on es pot veure el seu funcionament:

```
public class RoundsAdapter extends ArrayAdapter<GolfRound> {
	…
	public RoundsAdapter(Context context, int resource, List<GolfRound> items) {
	…
	}

	@Override
	public View getView(int position, View convertView, ViewGroup parent)
	{
		LinearLayout roundView;
		//Get the current object
		GolfRound round = getItem(position);

		//Inflate the view
		…
		//Get the text boxes from the list_rounds_item.xml file
		TextView roundTitle =(TextView)roundView.findViewById(R.id.roundTitle);
		TextView roundCourse =
					(TextView)roundView.findViewById(R.id.roundCourse);
		TextView roundDate =(TextView)roundView.findViewById(R.id.roundDate);

		//Assign the appropriate data from our alert object above
		roundTitle.setText(round.getTitle());
		roundCourse.setText(round.getGolf_course());
		roundDate.setText(round.getDate());

		return roundView;
	}

}
```

Amb l'adaptador definit i les funcions per obtenir les partides ja podem anar a l'activitat del llistat (ListRounds) i mostrar el llistat. Per fer-ho, haurem d'obtenir les partides amb la funció listrounds() que hem definit a Utils.java i assignar el llistat de partides que retorna a l'adaptdador RoundsAdaptador indicant el disseny a utilitzar (list_rounds_item.xml). Finalment, indicarem que el llistat que hem definit en el disseny de la pantalla mostri els elements que RoundsAdapter ha creat. A continuació indiquem el codi:

```
public class ListRounds extends Activity
{
 private ListView lv1;
 RoundsAdapter ra;
 ArrayList<GolfRound> rounds;
 Context c; Utils u;
 @Override
 public void onCreate(Bundle icicle)
 {
        super.onCreate(icicle);
        setContentView(R.layout.list_rounds);
        lv1=(ListView)findViewById(R.id.ListView01);
        c= this;
       //get rounds
       u = new Utils(this);
       rounds= u.getRounds();

       //Initialize our array adapter notice how it references the listitems.xml
layout
       ra = new RoundsAdapter(this, R.layout.list_rounds_item,rounds);

       //Set the above adapter as the adapter of choice for our list
       lv1.setAdapter(ra);

 }
```

En aquest punt ens trobem que ja tenim el llistat de partides i podem visualitzar-les per pantalla. La part que resta per fer es poder filtrar el llistat de partides per camp de golf.

La idea es mostrar un diàleg que ens permeti introduir el nom del camp i mentre anem escrivint ens mostri els camps que coincideixen amb el text escrit (funció d'autocomplete). El problema es que la funció d'autocompletar no la podem realitzar en un diàleg, sinó que l'hem de realitzar en una Activitat. Per sort, en Android podem incloure una activitat a dins d'un diàleg.

Per fer-ho primer crearem un nou objecte que anomenarem "CourseItem" que només contindrà el identificador i el nom del camp en el fitxer "CourseItem.java".

```
public class CourseItem {
   private String cid;
   private String name;


…

}
```

Després a Utils.java crearem una funció que ens retornarà un llistat d'objectes de tipus "CourseItem". Adjuntem només la capçalera de la funció:

```
/**
 *
 * @return ArrayList of User Rounds
 */
public ArrayList<CourseItem> getCourses(){…}
```

Un cop feta la funció crearem una nova activitat, que és la que ficarem a dins del diàleg de selecció del camp, i que ens realitzarà la tasca de suggerir els camps de golf que coincideixen amb el text que escriu l'usuari. Definirem aquesta nova activitat en el manifest i crearem el fitxer ListCourses.java que la implementarà.

El disseny de l'activitat el definirem a en el fitxer "res/layout/select_course.xml". Aquest fitxer únicament definirà el camp de text per indicar el camp de golf, que serà de tipus AutoCompleteTextView i un botó que usarem per indicar que ja hem escollit el camp.

A continuació ja podem anar a implementar l'activitat ListCourses. El funcionament d'un AutoCompleteTextView és molt semblant al d'una llista. Primer obtenim el llistat d'elements, en aquest cas camps de golf. Després creem un adaptador on anirem afegint els noms del camp de golf. En aquest cas podem utilitzar l'adaptador genèric ja que el llistat que apareixerà per pantalla amb els camps de golf coincidents no mostra cap tipus d'objecte, només el nom del camp. Finalment, haurem d'assignar el adaptador definit al camp de text de tipus AutoCompleteTextView.

```
public class ListCourses extends Activity
{
 ArrayList<CourseItem> courses;
 ArrayAdapter<String> adapter;
 AutoCompleteTextView textView;
 String cid="";
 Context c;

 @Override
 public void onCreate(Bundle icicle)
 {
        super.onCreate(icicle);
        setContentView(R.layout.select_course);
        c=this;
        Utils u = new Utils (this);
        courses = u.getCourses();
        adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line);

        for (int i = 0; i< courses.size();i++){
               adapter.add(courses.get(i).getName());
        }

        textView = (AutoCompleteTextView) findViewById(R.id.golf_course);
        textView.setAdapter(adapter);
…
```

El següent punt es definir el comportament del botó d'acceptar que indicarà que l'usuari ja ha seleccionat el camp. El que farem es comprovar que el text introduït correspon al nom d'un camp. En cas contrari, mostrarem un missatge d'error indicant a l'usuari que el nom no és vàlid. En el cas contrari, el nom del camp es vàlid, retornarem a l'activitat de llistar partides (ListRounds) el identificador numèric del camp de golf pel qual s'ha de filtrar el resultat del llistat. Amb la funció setResult() una activitat pot retornar un resultat a una altra. Després de retornar el valor finalitzarem l'activitat de seleccionar el camp de golf. A continuació mostrem la funció que retorna l'identificador del camp.

```
private void returnCourse(int resCode){
        Intent i;
     i=new Intent();
     Bundle b = new Bundle();
     b.putString("cid",cid);
     i.putExtras(b);
     setResult(resCode,i);
     finish();
 }
```

En l'activitat ListRounds haurem de llançar l'activitat ListCourses a dins d'un diàleg quan es seleccioni l'opció del menú per filtrar per camp de golf. Després s'haurà d'obtenir l'identificador

del camp que retorna per finalment, filtrar el contingut del llistat fent que només surtin les partides del camp indicat.

A continuació mostrem el codi de l'activitat ListRounds que llança l'activitat al seleccionar l'opció del menú utilitzant la funció d'Android startActivityForResult(), que permet llançar una activitat i esperar que retorni un valor. La funció onActivityResult() és la que s'encarrega de rebre el resultat i filtrar les partides pel camp de golf.

```
public boolean onOptionsItemSelected(MenuItem item) {
     switch (item.getItemId()) {
     case R.id.course_filter:
          Intent icourses= new Intent(ListRounds.this, ListCourses.class);
                startActivityForResult(icourses,0);
     }
     return false;
 }

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(resultCode == 0){
            Bundle b = data.getExtras();
            String cid= b.getString("cid");
            rounds= u.getRoundsbyCourse(cid);
            ra = new RoundsAdapter(this, R.layout.list_rounds_item,rounds);
            //Set the above adapter as the adapter of choice for our list
               lv1.setAdapter(ra);
    }
}
```

Després d'aquest llarg procés ja podem llistar activitats i filtrar pel camp de golf. La següent il·lustració mostra el funcionament del llistat.

Il·lustració 77 – Pantalla de llistar les partides d'usuari amb opció de filtrar per camp de golf

Mostrar el resultat d'una partida

Per mostrar el resultat d'una partida primer definirem una nova activitat al manifest i crearem el fitxer GolfScore.java que implementarà l'activitat.

Amb l'activitat ja creada, tornarem al llistat de partides per definir la l'acció de mostrar la puntuació d'una partida al seleccionar una partida del llista.

En el fitxer ListRounds.java definirem l'esdeveniment "OnItemClickListener" per a l'element de tipus llista que conté les partida. Aquest esdeveniment ens permetrà saber quin element del llistat s'ha seleccionat per a poder indicar a la nova activitat (GolfScore) de quina partida ens ha de mostrar la seva puntuació. Per passar valors d'una activitat a una altra utilitzarem la funció d'Android "putExtras()".

```
lv1.setOnItemClickListener(new OnItemClickListener() {
	@Override
	public void onItemClick(AdapterView<?> a, View v, int position, long id) {
		GolfRound round= (GolfRound)lv1.getItemAtPosition(position);

		Intent iscore= new Intent(c, GolfScore.class);
		Bundle bundle = new Bundle();
		bundle.putString("nid", round.getNid());
		iscore.putExtras(bundle);
		startActivity(iscore);
	}
});
```

Ara ja podem definir l'activitat que ens mostrarà el resultat de la partida. Per fer-ho ens ajudarem de l'aplicació MiniGolfScore tal com es va indicar en l'anterior iteració. D'aquesta aplicació ens quedarem amb els següents elements:

- **SheetView.java**: Aquesta classe s'encarrega de crear la pantalla amb la taula que conte les puntuacions de l'usuari. S'han realitzat les següents modificacions:
    - Modificació dels colors de la taula per adaptar-los a l'estil de l'aplicació
    - Modificació de la mida dels textos per una millor visualització dels resultat
    - Modificació de la doble línea per separar les files de d'informació del camp i de les columnes amb el nom del jugador i el total de les caselles amb els resultats

- **ScoreData.java**: Fitxer que conté la puntuació de la partida i el par del camp. Aquest fitxer s'ha modificat per permetre guardar tota la informació del camp de golf (par, handicap i longitud de cada forat).
- **dialog_edit_score**: Fitxer que conté el disseny de la pantalla que permet editar la informació d'un forat.
- També s'ha utilitzat part del codi de l'activitat principal de l'aplicació MiniGolfScore que servia per comunicar-se amb els elements de la pantalla. En concret, de les definicions dels esdeveniments que permeten:
    - Modificar la puntuació concreta d'un jugador en un forat
    - Desar les dades que es troben per pantalla per continuar després omplint la puntuació (ho utilitzarem en la pantalla de nova partida només)

Amb la incorporació d'aquests elements a l'activitat GolfScore els dos passo que hem se seguir per visualitzar la informació de la partida són:

1. Obtenir l'identificador de la partida que ens ha passat l'activitat del llistat de partides al seleccionar un element:

```
//get golf round id
//-----------------------------------------
Bundle bundle = this.getIntent().getExtras();
String snid = bundle.getString("nid");
```

2. Obtenir la informació de la partida i assignar-la al objecte "ScoreData" per indicar la puntuació que ha de sortir per pantalla. Això ho farem amb la funció setScoreData(ScoreData sd, String id_partida) que hem creat al fitxer Utils.java. Aquesta funció obté la puntuació de la partida indicada i la informació del camp de golf assignant-la al objecte "ScoreData" que la classe SheetView s'encarregarà de mostrar per pantalla:

```
u= new Utils(GolfScore.this);

u.setScoreData(mScoreData, snid);
```

A l'aplicació MinigolfScore quan es clicava a sobre del par d'un forat del camp s'obria un diàleg que permetia editar-lo. En la nostra aplicació substituirem aquest diàleg per un altre

que indicarà el par, el handicap i la longitud del forat ja que nosaltres disposem de la informació del camp. El disseny d'aquest diàleg es troba a "res/layout/dialog_hole_info.xml.

```
case DIALOG_HOLE_INFO:
        //Get the text boxes from the dialog_hole_info.xml file
        TextView par_info =(TextView)mViewHoleInfo.findViewById(R.id.hole_par);
        TextView handicap_info =
                        (TextView)mViewHoleInfo.findViewById(R.id.hole_handicap);
        TextView length_info =
                        (TextView)mViewHoleInfo.findViewById(R.id.hole_length);

        //Assign the appropriate data from our alert object above
        par_info.setText(String.format("%d", mScoreData.getPar(mEditHole)));
        handicap_info.setText(String.format("%d",mScoreData.getHandicap(mEditHole)));
        length_info.setText(String.format("%dm",mScoreData.getLength(mEditHole)));

        mDialogTitle = String.format("Information for hole %d", mEditHole + 1);
        ad.setTitle(mDialogTitle);
        break;
```

Un cop realitzat el procés d'assignar la puntuació de la partida, la informació del camp i definir el diàleg amb la informació de cada forat ja podem realitzar el procés complet per visualitzar la informació de la partida.

Finalment, amb tota la funcionalitat creada podem mostrar les pantalles de l'aplicació que mostren la puntuació d'una partida amb la informació de cada forat del camp.

Il·lustració 78 – Pantalla de visualització de la puntuació d'una partida de golf

11 Iteració 8

11.1 Llista d'activitats de la iteració

11.1.1 Creació de partides de golf

1- Permetre la creació d'una nova partida de golf
2- Escollir el camp de golf de la partida
3- Mostrar la targeta de la partida buida preparada per afegir la puntuació
4- Permetre selecció de jugadors del portal en Drupal
5- Mantenir la informació de la nova fins que es decideixi esborrar-la o enviar al servidor.
6- Enviar les dades de la partida al servidor

11.1.2 Actualització de partides existents

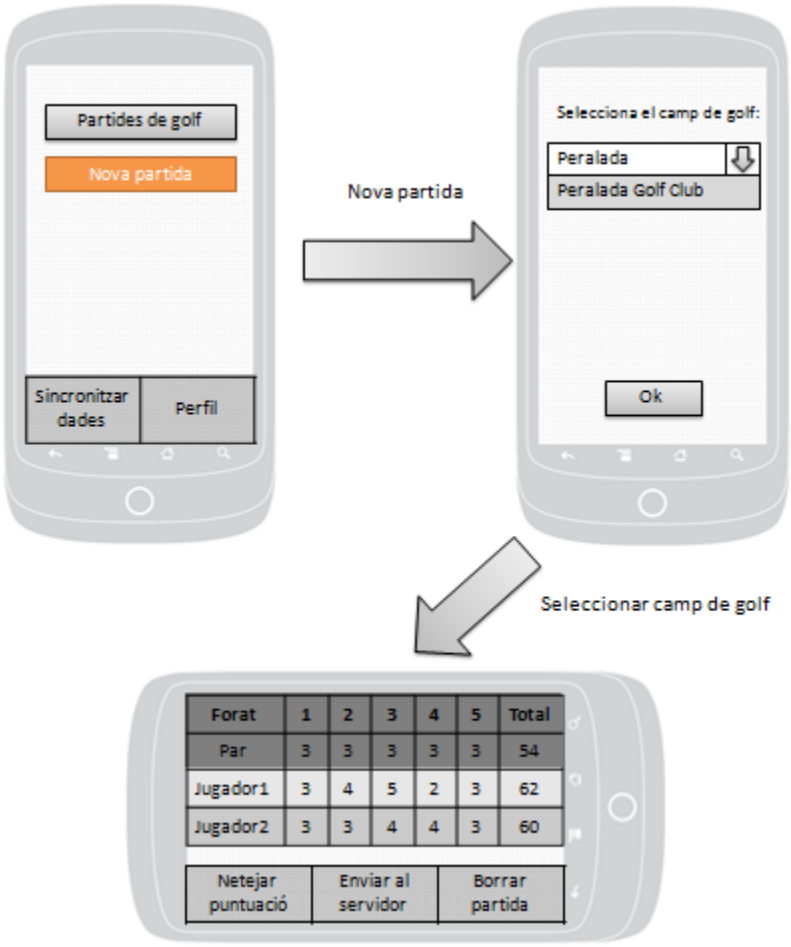
1- Permetre editar el resultat d'una partida existent, incloent els noms dels jugadors.
2- Permetre a l'usuari actualitzar les dades de la partida.

11.2 Anàlisi i disseny de les funcionalitats

11.2.1 Creació de partides de golf

Funcionalitat que permetrà a l'usuari crear noves partides de golf que es guardaran a la base de dades de Drupal.

11.2.1.1 Maqueta de la funcionalitat



Il·lustració 79 – Maqueta de la creació d'una partida de golf

11.2.1.2 Especificació detallada

A continuació farem una explicació detallada de com s'ha de comportar l'aplicació:

- Al clicar el botó “Nova partida” de la pantalla principal es mostrarà a l’usuari una pantalla per seleccionar el camp de golf de la partida.
- Al seleccionar el camp l’aplicació mostrarà la targeta de la partida buida, únicament amb la informació dels forats del camp.
- La pantalla que mostrarà la targeta definirà un menú amb les opcions de reiniciar la partida (esborrar les puntuacions introduïdes) , esborrar la partida (tornant a la pantalla principal) i enviar al servidor.
- L’aplicació permetrà guardar la puntuació inserida al sortir de la pantalla permetent continuar després afegint la resta de la puntuació.
- Al clicar en una casella reservada a guardar la puntuació d’un jugador en un forat es mostrarà un diàleg que permetrà indicar la puntuació de la casella.
- Al clicar en el par del forat veurem la informació detallada del forat en un diàleg.
- Al clicar damunt la casella amb el nom del jugador s’obrirà un diàleg per editar el nom. L’aplicació únicament permetrà indicar jugadors del sistema.

11.2.1.3 Disseny de la funcionalitat

El primer pas serà definir una nova activitat dins el manifest per a la pantalla de nova partida i crear el fitxer NewGolfScore que la implementarà:

```
<activity android:name=".NewGolfScore"
          android:label="@string/app_name"
            android:configChanges="keyboardHidden|orientation">
    <intent-filter>
        <category android:name="com.kbza.drupalgolf.NEW_GOLF_CARD" />
    </intent-filter>
</activity>
```

Amb l’activitat creada anirem a l’activitat principal DrupalGolf i definirem el codi que s’executarà quan es prem el botó de “Nova partida” que mostrarà en un diàleg l’activitat que permet seleccionar un camp de golf definida a la iteració anterior en la funcionalitat de llistar partides o si existeix una partida ja començada ens mostrarà directament la pantalla de la nova partida. Per distingir entre els dos casos es mirarà si existeix el fitxer que conté la puntuació de la partida que s’està jugant. Si no existeix es que volem crear una partida nova.

```
final Button buttonNewRound = (Button) findViewById(R.id.btn_new_round);
buttonNewRound.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
       /* Load new round activity.
        */
       String[] files = fileList();
       boolean exists =false;
       for (int i=0; i<files.length;i++){
              if (files[i].equals(NewGolfScore.SAVE_FILENAME)) exists = true;
       }
       if (!exists){
              Intent icourses= new Intent(DrupalGolf.this, ListCourses.class);
              startActivityForResult(icourses,0);
       }else {
              Intent iscore= new Intent(c, NewGolfScore.class);
              Bundle bundle = new Bundle();
              bundle.putString("cid", NewGolfScore.CONTINUE_GAME);
              iscore.putExtras(bundle);
              startActivity(iscore);
       }
    }
});

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(resultCode == 0){
              Bundle b = data.getExtras();
              Intent iscore= new Intent(c, NewGolfScore.class);
              Bundle bundle = new Bundle();
              bundle.putString("cid", b.getString("cid"));
              iscore.putExtras(bundle);
              startActivity(iscore);
    }
}
```

En el codi es pot veure que si no existeix el fitxer es crida a l'activitat "ListCourses" i quan l'usuari indica el camp de golf a la funció `onActivityResult()` llancem l'activitat NewGolfScore indicant-li el identificador del camp de golf per poder crear la nova partida. En cas de tenir un partida començada (existeix el fitxer) executarem directament l'activitat NewGolfScore indicant un valor constant per que l'activitat pugui saber que s'ha de continuar la partida enlloc de crear una de nova. El fitxer on es guarda temporalment la partida que s'està jugant el crearem a l'activitat de nova partida.

Pel codi de nova partida partirem de l'activitat de GolfScore (visualització de partida) però modificant que el paràmetre que rebrà serà el identificador d'un camp de golf o una constant indicant que s'ha de continuar la partida existent enlloc de l'identificador de la partida.

En el cas de crear una nova partida indicarem a la pantalla que inicialitzi la partida amb les dades del camp indicat.

```
//get golf course
//-----------------------------------------
Bundle bundle = this.getIntent().getExtras();
cid = bundle.getString("cid");
if (!cid.equals(CONTINUE_GAME)){
       u.setCourseData(mScoreData, cid);
}
```

En canvi si estem continuant una partida en la funció onResume() que s'executarà quan una pantalla es mostra carregarem les dades del fitxer mitjançant la funció loadFromFile() definida al fitxer ScoreData.java

```
@Override
protected void onResume() {
       super.onResume();

       // Restore saved settings if exists SAVE_FILENAME
       String[] files = fileList();
       boolean exists =false;
       for (int i=0; i<files.length;i++){
             if (files[i].equals(NewGolfScore.SAVE_FILENAME)) exists = true;
       }
       if (exists && !changePlayer){
        mScoreData.loadFromFile(this, SAVE_FILENAME);
          …
       }
}
```

Acabem de veure com es càrrega el fitxer però, abans l'aplicació l'ha d'haver creat. El fitxer que conté les dades de la partida es guarda cada cop que es pausa l'activitat (això pot ser degut a que per exemple l'usuari canvia d'aplicació) mitjançant la funció d'Android onPause().

```
/**
 * Save settings to the specified file
 */
private void saveSettings(String filename) {
	mScoreData.setSavedScoreRelative(mScoreSheet.getScoreRelative());
	mScoreData.setSavedSelPlayer(mScoreSheet.getSelectedPlayer());
	mScoreData.setSavedSelHole(mScoreSheet.getSelectedHole());
	mScoreData.saveToFile(this, filename);
}

/**
 * Activity is being paused.  It may be killed after onPause() returns.
 * @see android.app.Activity#onPause()
 */
@Override
protected void onPause() {
	super.onPause();
	if (saveClose) saveSettings(SAVE_FILENAME);
	else saveClose = true;
}
```

Les funcions loadFromFile i saveToFile han sigut modificades respecte les originals que s'utilitzaven a l'aplicació MinigolfScore per permetre poder guardar la informació del camp de golf.

En aquest punt tenim definida la pantalla i la càrrega de les dades en cas de continuar la partida existent. La feina que ens queda es definir les accions del menú així com definir la pantalla d'edició del nom de jugador.

Per definir la pantalla d'edició del nom d'un jugador crearem el mateix procés que per a seleccionar un camp de golf. Es a dir, crear una activitat que es mostrarà en un diàleg i permetrà sugerir els jugadors del sistema que coincideixen amb el text escrit per l'usuari. No tornarem a detallar el procés ja que pràcticament és idèntic que el creat pels camp de golf.

Les tres opcions del menú (reiniciar la puntuació, enviar la partida al servidor i esborrar la partida) les definirem en el fitxer "res/menu/new_golf_score_menu.xml".

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
       <item android:title="Clear scores" android:id="@+id/clear"
              android:icon="@android:drawable/ic_menu_close_clear_cancel"
              android:alphabeticShortcut="c" />
       <item android:titleCondensed="Send to server" android:id="@+id/save"
              android:icon="@android:drawable/ic_menu_send"
              android:alphabeticShortcut="s" />

       <item android:titleCondensed="Delete round" android:id="@+id/delete"
              android:icon="@android:drawable/ic_menu_delete"
              android:alphabeticShortcut="d" />
</menu>
```

En el cas de prémer l'opció de reiniciar la puntuació d'una partida, l'aplicació mostrarà un diàleg que ens demanarà conformitat per realitzar l'acció. En cas d'acceptar esborrar la puntuació introduïda. En cas contrari no realitzarà cap acció.

```
…
case DIALOG_CONFIRM_CLEAR:
    return new AlertDialog.Builder(NewGolfScore.this)
        .setIcon(android.R.drawable.ic_menu_close_clear_cancel)
        .setTitle(R.string.dialog_confirm_clear_title)
        .setPositiveButton(R.string.dialog_yes, new DialogInterface.OnClickListener()
       {
            public void onClick(DialogInterface dialog, int whichButton) {
              // User clicked Yes; clear scores.
              mScoreData.resetScores();
              mScoreData.resetPlayerNames();
              mScoreSheet.setSelectedHole(0);
              mScoreSheet.setSelectedPlayer(0);
            }
         })
        .setNegativeButton(R.string.dialog_no, new DialogInterface.OnClickListener()
       {
            public void onClick(DialogInterface dialog, int whichButton) {
              // User clicked Cancel; do nothing
            }
         })
        .create();
…
```

En el cas d'esborrar la partida l'acció a realitzar serà eliminar el fitxer on es va guardant la nova partida i finalitzar l'activitat tornant a la pantalla principal. S'utilitza la variable "saveClose" per indicar que, al parar l'activitat, no torni a crear el fitxer (per defecte es guarda per a poder disposar del contingut quan l'usuari tanca l'aplicació però no vol perdre la puntuació). També s'indicarà un diàleg que demanarà confirmació a l'usuari abans d'esborrar.

```
…
case DIALOG_CONFIRM_DELETE:
       return new AlertDialog.Builder(NewGolfScore.this)
    .setIcon(android.R.drawable.ic_menu_close_clear_cancel)
    .setTitle(R.string.dialog_confirm_delete_title)
    .setPositiveButton(R.string.dialog_yes, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
              // User clicked Yes; delete round.
              deleteFile(SAVE_FILENAME);
              saveClose= false;
              finish();
        }
    })
    .setNegativeButton(R.string.dialog_no, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
              // User clicked Cancel; do nothing
        }
    })
    .create();
…
```

Com a última opció del menú, tenim l'opció d'enviar la partida al servidor. Per a poder fer-ho primer definirem en la interfície Client.java i el client JSONServerClient.java la funció que és comunicarà amb Drupal per poder guardar partides de golf.

```
@Override
public String newRound(String title, String course, String player1, String player2,
String player3, String player4)
              throws ServiceNotAvailableException {
       String nid = null;
       try{
              BasicNameValuePair[] parameters = new BasicNameValuePair[6];

       parameters[0] = new BasicNameValuePair("title",
                           "\""+URLEncoder.encode(title,"UTF-8")+"\"");
       parameters[1] = new BasicNameValuePair("course", "\""+course+"\"");
       parameters[2] = new BasicNameValuePair("player1","\""+player1+"\"");
       parameters[3] = new BasicNameValuePair("player2", "\""+player2+"\"");
       parameters[4] = new BasicNameValuePair("player3", "\""+player3+"\"");
       parameters[5] = new BasicNameValuePair("player4", "\""+player4+"\"");
       nid=call("newround.new", parameters);
       }catch(UnsupportedEncodingException e){}

       return nid;
}
```

Un cop definida al fitxer Utils.java crearem la funció que utilitzarà el mètode que acabem de definir. Aquesta funció agafarà el resultat de la partida i el passarà al format que requereix el mètode. Recordem que en la iteració 5 es va definir el servei de manera que cada jugador s'havia de passar de la següent forma: "nom_jugador1, puntuació_forat1,…, puntuació_forat_18".

```
public int newRound(ScoreData sd, String title) {
	int nid= 0;
	String player1 ="";String player2 =""; String player3 =""; String player4 ="";
	String cid= sd.getCourse().getCid();
	if (getPlayerByName(sd.getPlayerName(0))!=null){
		player1+=sd.getPlayerName(0);
		for (int h=0; h<DrupalGolf.NUM_HOLES;h++)
	player1+=","+sd.getScore(0,h);
	}
	…
	if (getPlayerByName(sd.getPlayerName(3))!=null){
		player4+=sd.getPlayerName(0);
		for (int h=0; h<DrupalGolf.NUM_HOLES;h++)
	player4+=","+sd.getScore(3,h);
	}
	try{
		String res= client.newRound(title, "[nid:"+cid+"]", player1, player2,
					player3, player4);
		JSONObject jso = new JSONObject(res);

		nid = jso.getJSONObject("#data").getInt("nid");

	}catch (ServiceNotAvailableException e) {
		e.printStackTrace();
	}catch (JSONException e) {
		e.printStackTrace();
	}
	return nid;
}
```

Tornant a l'activitat NewGolfScore definirem el comportament al prémer l'opció d'enviar partida al servidor. En primer terme un diàleg ens permetrà indicar el nom que li volem donar a la partida, per posteriorment, comprovar que la puntuació introduïda es correcta (si no es correcta mostrarem un missatge d'error) i enviar la partida al servidor. Un cop guardada la partida mostrarem un missatge a l'usuari indicant que la partida s'ha creat correctament i actualitzarem el fitxer que conté les partides del usuari. Un cop guardat també esborrarem el fitxer amb el contingut de la partida (ja que ara es troba guardada en el Drupal) i tancarem l'activitat.

```
case DIALOG_NEW_ROUND:
        return new AlertDialog.Builder(NewGolfScore.this)
        .setMessage("Round created with nid: "+String.format("%d",nid))
              .setPositiveButton(R.string.dialog_ok, new
                    DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
              deleteFile(SAVE_FILENAME);
              saveClose= false;
              u.userRounds();
              finish();
        }
       })
        .create();
case DIALOG_TITLE_ROUND:
       return new AlertDialog.Builder(NewGolfScore.this)
       .setTitle(mDialogTitle)
       .setView(mViewRoundTitle)
       .setPositiveButton(R.string.dialog_ok, new DialogInterface.OnClickListener() {
             public void onClick(DialogInterface dialog, int whichButton) {
              // User clicked OK, so set score.
                    EditText t = (EditText)
                          mViewRoundTitle.findViewById(R.id.round_title_edit);
                    AlertDialog.Builder adb=
                    new AlertDialog.Builder(NewGolfScore.this);
                    // Log.d("MiniGolfScore", "E-MAIL MENU");
                    if (u.checkScore(mScoreData)){
                          u.login();
                          nid=u.newRound(mScoreData,t.getText().toString());
                          if (nid!=0){
                                showDialog(DIALOG_NEW_ROUND);

                          }else{
                                adb.setMessage(R.string.error_new_round);
                                adb.setPositiveButton(R.string.dialog_cancel, null);
                                adb.show();
                          }
                    }else{
                          adb.setMessage(R.string.error_invalid_score);
                          adb.setPositiveButton(R.string.dialog_cancel, null);
                          adb.show();
                    }
                }
            })
            .setNegativeButton(R.string.dialog_cancel, new
             DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int whichButton) {
                    // User clicked cancel; do nothing}})
            .create();
```

Amb tot el codi desenvolupat ja ens trobem en disposició de poder crear partides des de l'aplicació client en Android. A continuació mostrem les diferent pantalles i diàlegs relacionats amb la funcionalitat.

Il·lustració 80 – Pantalles i diàlegs relacionats amb la creació d'una nova partida

11.2.2 Actualització de partides existents

11.2.2.1 Maqueta de la funcionalitat

Il·lustració 81 – Maqueta de la pantalla de visualitzar partida amb opció d'actualitzar puntuació

11.2.2.2 Especificació detallada

A continuació farem una explicació detallada de com s'ha de comportar l'aplicació:

- En la pantalla de visualització d'una partida és permetrà a l'usuari modificar la puntuació i els noms del jugadors.
- Com a opció del menú es permetrà actualitzar la puntuació de la partida en el servidor. Es demanarà confirmació a l'usuari.
- Es mostrarà missatge indicant que la partida s'ha actualitzat correctament.

11.2.2.3 Disseny de la funcionalitat

Com a base per a realitzar la funcionalitat tenim la pantalla de visualització de la partida que ja ens permet modificar la puntuació de cada casella de la targeta de golf. El que s'ha d'afegir és el diàleg de modificació del nom d'un jugador de la mateixa forma que s'ha per a la funcionalitat anterior (crear nova partida).

Un cop feta aquesta tasca definirem l'opció del menú que ens permetrà actualitzar la partida de golf en el servidor Drupal en el fitxer "res/menu/golf_score_menu.xml".

```xml
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
       <item android:titleCondensed="Update in server" android:id="@+id/update"
              android:icon="@android:drawable/ic_menu_send"
              android:alphabeticShortcut="s" />
</menu>
```

A continuació definirem en la interfície Client.java i en la classe JSONServerClient.java que la implementa la funció que és comunicarà amb Drupal i actualitzarà la informació d'una partida de golf.

```
@Override
public String updateRound(String nid, String title, String course, String player1,
String player2, String player3, String player4) throws ServiceNotAvailableException {

       try{
              BasicNameValuePair[] parameters = new BasicNameValuePair[7];

       parameters[0] = new BasicNameValuePair("nid", nid);
       parameters[1] = new BasicNameValuePair("title",
                                   "\""+URLEncoder.encode(title,"UTF-8")+"\"");
       parameters[2] = new BasicNameValuePair("course", "\""+course+"\"");
       parameters[3] = new BasicNameValuePair("player1","\""+player1+"\"");
       parameters[4] = new BasicNameValuePair("player2", "\""+player2+"\"");
       parameters[5] = new BasicNameValuePair("player3", "\""+player3+"\"");
       parameters[6] = new BasicNameValuePair("player4", "\""+player4+"\"");
       nid=call("newround.update", parameters);
       }catch(UnsupportedEncodingException e){}

       return nid;
}
```

En el fitxer Utils.java crearem una funció que utilitzarà el mètode updateRound() que acabem de definir.

```
public int updateRound(ScoreData sd) {
	int nid= 0;
	String player1=""; String player2=""; String player3=""; String player4="";
	String cid= sd.getCourse().getCid();
	if (getPlayerByName(sd.getPlayerName(0))!=null){
		player1+=sd.getPlayerName(0);
		for (int h=0; h<DrupalGolf.NUM_HOLES;h++) player1+=","+sd.getScore(0,h);
	}
	…
	if (getPlayerByName(sd.getPlayerName(3))!=null){
		player4+=sd.getPlayerName(0);
		for (int h=0; h<DrupalGolf.NUM_HOLES;h++) player4+=","+sd.getScore(3,h);
	}
	try{
		String res= client.updateRound(sd.getNid(),sd.getTitle(),
				"[nid:"+cid+"]", player1, player2, player3, player4);
		JSONObject jso = new JSONObject(res);
		nid = jso.getJSONObject("#data").getInt("nid");

	}catch (ServiceNotAvailableException e) {
		e.printStackTrace();
	}catch (JSONException e) {
		e.printStackTrace();
	}

	return nid;
}
```

En l'activitat GolfScore (on mostrem la partida) definirem el comportament al seleccionar l'opció del menú d'actualitzar la partida. El comportament consistirà en mostrar un diàleg de confirmació a l'usuari. En cas d'acceptar es comprovarà que la puntuació sigui correcta (que no estigui cap casella sense informació) i s'actualitzarà la partida en el servidor. Després es mostrarà un diàleg indicant que el procés s'ha realitzat amb èxit per finalment actualitzar el fitxer amb les partides d'usuari des de el servidor i finalitzar l'activitat. Al finalitzar l'activitat es tornarà a la pantalla amb el llistat de partides de l'usuari.

```
case DIALOG_CONFIRM_UPDATE:
    return new AlertDialog.Builder(GolfScore.this)
    .setTitle(R.string.dialog_confirm_update)
    .setPositiveButton(R.string.dialog_yes, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
              // User clicked Yes; update scores.
              AlertDialog.Builder adb=new AlertDialog.Builder(GolfScore.this);
              // Log.d("MiniGolfScore", "E-MAIL MENU");
              if (u.checkScore(mScoreData)){
                    u.login();
                    nid=u.updateRound(mScoreData);
                    if (nid!=0){
                          showDialog(DIALOG_UPDATE_ROUND);

                    }else{
                          adb.setMessage(R.string.error_new_round);
                          adb.setPositiveButton(R.string.dialog_cancel, null);
                          adb.show();
                    }
              }else{
                    adb.setMessage(R.string.error_invalid_score);
                    adb.setPositiveButton(R.string.dialog_cancel, null);
                    adb.show();
              }
        }
    })
    .setNegativeButton(R.string.dialog_no, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
              // User clicked Cancel; do nothing
        }
    })
    .create();
```

Per finalitzar la funcionalitat mostrem les diferents pantalles i diàlegs relacionats amb el procés d'actualització d'una partida.

Il·lustració 82 – Pantalles i diàlegs relacionats amb la creació d'una nova partida

12 Anàlisi final

Al llarg d'aquest document s'ha vist com s'ha realitzat el desenvolupament del projecte. Un cop finalitzat el projecte podem extreure les següents conclusions que seran mostrades en els següents punts.

Ens centrarem en els problemes tècnics i les dificultats sorgides al llarg del projecte, el temps final real que ens ha ocupat realitzar totes les tasques definides, els cost final del projecte, definir futures ampliació per finalment acabar amb una valoració personal on indicarem les nostres impressions.

12.1 Problemes tècnics

Durant la realització del projecte han sorgit diferents problemes tècnics relacionats amb incompatibilitat de mòduls del Drupal, problemes degut a la manca de coneixement suficient (aspecte normal al desenvolupar en una plataforma per primer cop), problemes al comunicar el client amb el servidor degut a la llibreria utilitzada,etc.

S'ha intentat que aquests problemes afectin de forma mínima al calendari marcat pel desenvolupament fent que totes les iteracions s'hagin assolit amb èxit i no s'hagin descartat més funcionalitats de les que es van descartar en el període de plantejament de les funcionalitats a assolir en el desenvolupament.

Un cop realitzada la feina podem valor positivament l'aparició d'aquests problemes ja que realment quan més s'aprèn es quan ens topem amb problemes que hem de solucionar.

S'ha d'agrair la gran comunitat de desenvolupadors de Drupal que fan possible la solució de molts dubtes de desenvolupadors més inexperts així com donar resposta a problemes derivats de limitacions dels mòduls. En el cas d'Android també hi ha molta informació. La diferencia és que, a diferencia de Drupal on a la seva web podem trobar pràcticament tota la informació relacionada als problemes que poden sorgir en un desenvolupament, en Android la seva web

serveix més de referència per als desenvolupadors i la informació de com solucionar problemes o realitzar diverses funcionalitats es troba més dispersa.

12.2 Temps final

En l'apartat 2.2 d'aquest document es va proposar una estimació de les hores necessàries per cada funcionalitat. Aquesta estimació es basava principalment en la dificultat assignada a priori tenint en compte els coneixements inicials sobre la plataforma. Un cop realitzada cada funcionalitat, s'ha fet un càlcul de les hores que emprades en finalitzar la funcionalitat. El resultat queda exposat en les següents taules:

Portal Drupal:

Funcionalitat	Tems real de desenvolupament
Gestió d'activitats	25 hores
Vista d'activitats agrupades per tipus	30 hores
Apuntar-se a les activitats	25 hores
Vista de les activitats on participo	18 hores
Vista de les activitats que gestiono	20 hores
Gestió de partides de golf	25 hores
Gestió de camps de golf	35 hores
Templates per visualitzar activitats, partides de golf i camps de golf	30 hores
Vista de les partides de l'usuari	20 hores
Gestió d'estadístiques de les partides de golf	65 hores

Creació del perfil d'usuari	28 hores
Exportació de serveis per aplicacions externes	65 hores
Unificació projectes paral·lels Drupal	20 hores
TOTAL	406 hores

Aplicació per a dispositius mòbils Android:

Funcionalitat	**Estimació esforç de programació**
Login/Logout usuari	30 hores
Sincronització de les dades	60 hores
Visualització de partides	45 hores
Actualització de partides existents	20 hores
Creació de noves partides	50 hores
TOTAL	205 hores

Les hores dedicades que es van preveure per l'estudi de la plataforma Drupal així com la posada en funcionamet del sistema i l'estudi de la plataforma Android més la creació de l'esquelet de l'aplicació queden intactes. Recordem que es dividien de la següent forma

- **100 hores** per a l'estudi del CMS Drupal i posar en funcionament la plataforma existent
- **80 hores** per l'estudi de la plataforma Android, més la creació esquelet de l'aplicació i proves de l'aplicació en dispositius reals:

Per tant el temps total dedicat és de:

	Hores
Estudi CMS Drupal	100 hores
Desenvolupaments Drupal	406 hores
Estudi plataforma Android	80 hores
Desenvolupament Android	205 hores
TOTAL	**791 hores**

El temps previst inicialment era de 810 hores, per tant s'ha trigat un 2.35% menys del temps estimat. Aquest desviament es pot considerar pràcticament nul.

Normalment, en un cas amb client real, l'estimació inicial en hores del projecte hagués estat una mica més optimista i agressiva, reduint força el nombre d'hores per oferir un preu més competent i poder captar així el client. En el nostre cas hem pogut optar per una estimació més conservadora donant més marge de maniobra de cara a afrontar els problemes sorgits durant el desenvolupament.

12.3 Cost final

En el apartat anterior s'ha comprovat que s'ha trigat menys hores de les previstes inicialment amb el client.

En un cas real el client ha d'acceptar el pressupost i com que la desviació és mínima no variarem el cost final del desenvolupament. Per tant, recuperant el preu definit a l'apartat 2.6, el cost del projecte es de **17.344,20€.**

12.4 Futures ampliacions

S'han definit possibles ampliacions si es volgués continuar amb el desenvolupament la plataforma. A continuació trobem un llistat que indica les més importants:

Portal Drupal

- Millorar els formularis de creació de partides de golf i camps de golf per facilitar la tasca de creació de contingut tant a administradors com usuaris
- Ampliar la gestió de partides de golf per poder organitzar tornejos de golf.
- Ampliar la temàtica de les activitats que es poden definir en el portal.
- La part del portal dedicada als productes no es troba pràcticament desenvolupada. Es podria muntar una botiga online amb els productes així com una millor classificació de cara al usuari.
- Es podria crear un sistema de localització de la publicitat fent que es mostri publicitat que sigui propera, per exemple, al codi postal de l'usuari o a la informació que s'està mostrant (ja sigui una casa rural, una activitat o un producte).
- Adaptar l'estil del portal quan sigui visitat per un dispositiu amb menys resolució com poden ser els dispositius mòbils.
- Permetre integració amb xarxes socials com Facebook o Twitter així com portals webs com flikr per permetre veure fotografies que els usuaris vulguin mostrar.

Client Android

- Permetre donar-se d'alta en el sistema via el client Android.
- Es podrien ampliar els continguts que mostrem del portal com, per exemple, llogar cases rurals des del client basat en Android.
- Es podria mostrar via Gogle maps les cases rurals o la ubicació on es realitzen activitats que es troben a prop de la ubicació de l'usuari en aquell precís instant.

12.5 Conclusió final

Amb el projecte finalitzat i tots els requisits i funcionalitats definits al inici del desenvolupament assolits podem extreure les següents conclusions personals.

Durant la realització del projecte hem aprés la importància com portar a terme un desenvolupament de mida considerable, definint clarament les funcionalitats i els objectius a assolir.

S'ha aprés els funcionament de dos plataformes obertes, Drupal i Android, referents cadascuna en el seu segment (Drupal en gestors de contingut i Android com a sistema operatiu i plataforma de creació d'aplicacions per a dispositius mòbils) que donen valor afegit al desenvolupador i que espero hem siguin d'utilitat en el mon laboral que m'espera.

Indirectament, s'ha millorat el nivell de programació, tant de PHP gràcies a Drupal com de Java gràcies a Android. No menys important ha sigut aprendre el format de declaració d'objectes JSON i l'ús de la potent llibreria jQuery. De fet, ja he pogut aplicar el coneixement adquirit en aquestes dues últimes en altres projectes.

També ens quedarem amb el que hem aprés de la metodologia àgil Scrum que, personalment he constatat, permet al client un millor seguiment del procés de creació del software al disposar al final de cada iteració de funcionalitats tangibles on poder valorar si s'està assolint amb èxit o no el projecte.

Per realitzar aquest projecte hem aplicat coneixements adquirits al llarg de la carrera. Especialment de les assignatures d'enginyeria del software que et permeten definir molt millor els requeriments, funcionalitats i comportament d'una aplicació, de les assignatures relacionades amb les bases de dades que han permès que interaccionar amb una base de dades sigui un procediment pràcticament trivial. També donar gràcies a les assignatures de programació que han permès que el Java sigui una facilitat en lloc d'un impediment a l'hora de treballar en Android i a les assignatures relacionades amb projectes i gestió de projectes, especialment PXC on es poden realitzar projectes de similars característiques.

Finalment, podem dir que aquest projecte ha satisfet les expectatives dels participants tot deixant una porta oberta a futures ampliacions de la plataforma.

13 Bibliografia

Llibres:

Using Drupal, Angela Byron, Addisson Berry, Nathan Haug, Jeff Eaton, James Walker i Jeff Robins – 2009

Pro Drupal Development 2nd Edition, John K. VanDyk, Matt Westgate – 2009

Android Application Development, Rick Rogers, John Lombardo, Zigurd Mednieks & Blake Meike – 2009

Publicacions web:

Drupal – www.drupal.org Pàgina oficial de Drupal amb mòduls, forums, versions del gestor,...

Android – http://developer.android.com/index.html Pagina oficial amb manual d'instalació i referencia de les principals llibreries del sistema

Android People – http://www.androidpeople.com/ Pàgina amb informació útil i tutorials d'Android

PHP – www.php.net Pàgina oficial de PHP

Json – http://json.org Pagina oficial de JSON

W3Schools – www.w3schools.com Tutorials per aprendre HTML i CSS

jQuery – http://jquery.com/ Pàgina oficial de jQuery amb informació i tutorials

DrupalCamp Barcelona - http://2010.drupalcamp.es/ - Campu sobre Drupal realitzat a Barcelona el Febrer del 2010. S'ha assistit a algunes de les conferències en qualitat d'oient.